VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Konvoluční neuronové sítě pro klasifikaci písma

Convolution Neural Networks for Script Classification

2019 Adam Podlas

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Adam Podlas**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Konvoluční neuronové sítě pro klasifikaci písma
Convolution Neural Networks for Script Classification

Jazyk vypracování: čeština

Zásady pro vypracování:

Hlavním cílem práce bude detekovat a správně klasifikovat 3 druhy asijských písem (korejské, čínské, japonské) na obrázcích za pomocí konvolučních neuronových sítí. Předpokládá se, že jednotlivá písma mohou být společně v jednom snímku a je třeba detekovat jejich pozici a druh.

Práce bude obsahovat:
1. Přehled použití hlubokých neuronových sítí pro analýzu obrazu.
2. Popis vybraných algoritmů a aplikací konvolučních sítí.
3. Návrh implementace a implementaci vybraných metod.
4. Popis generování datasetů pro učení a testování.
5. Porovnání výsledků metod.

Seznam doporučené odborné literatury:

[1] Goodfellow, Ian, Bengio, Yoshua and Courville, Aaron. Deep Learning, MIT Press, 2016
[2] LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner "Gradient-based learning applied to document recognition". Proceedings of the IEEE. 86 (11): 2278–2324.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jan Platoš, Ph.D.**

Datum zadání: 01.09.2018

Datum odevzdání: 30.04.2019

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 .....

Rád bych na tomto místě poděkoval vedoucímu mé diplomové práce doc. Ing. Janu Platošovi, Ph.D. za rady a konzultace při tvorbě této práce.

Abstrakt

Tato práce se zabývá možností použití konvolučních neuronových sítí k vizuálnímu rozpoznávání asijských jazyků (čínštiny, japonštiny a korejštiny) podle podoby jejich znaků. Hlavním cílem je vytvořit model schopný lokalizace a klasifikace textů v těchto jazycích v přirozených obrázcích. V rámci práce je navrhnut a implementován způsob generování syntetických dat pro zvýšení efektivity výsledného modelu. Dále tato práce obsahuje experimenty s různými architekturami, metodami učení a konfiguracemi hyperparametrů modelů za účelem nalezení optimálního řešení.

Klíčová slova: hluboké učení, konvoluční neuronové sítě, detekce objektů, detekce textu

Abstract

This diploma thesis explores possibilities of using convolutional neural networks for visual recognition of asian languages (chinese, japanese and korean) by the appearance of it's characters. The main goal of this work is to create a model for localization and classification of these languages' text in in natural scene images. The work contains design and implementation of a synthetic data generator for improving the resulting model. There are also experiments with different architectures, learning methods and hyperparameters configurations with the goal to find an optimal solution.

Key Words: deep learning, convolutional neural networks, object detection, text detection

Obsah

Seznam použitých zkratek a symbolů

mAP	–	Mean Average Precision
OCR	–	Optical Character Recognition
ReLU	–	Rectified Linear Unit
SVM	–	Support Vector Machine
RoI	–	Region of Interest
NMS	–	Non-Maximum Suppression
FPS	–	Frames Per Second
RPN	–	Region Proposal Network
FPN	–	Feature Pyramid Network
RoI	–	Region of Interest

Seznam obrázků

Seznam tabulek

1 Úvod

Konvoluční neuronové sítě v současné době dominují v nejrůznějších disciplínách spadajících do okruhu tzv. počítačového vidění. Metody založené právě na konvolučních sítích dosahuji jednoznačně nejlepších výsledků jak v klasifikaci, tak v detekci objektů. Cílem této práce je vytvořit model schopný detekce a klasifikace nejpoužívanějších asijských písem.

První část práce obsahuje analýzu problému detekce a identifikace textu v obrázku. Jedná se o rozbor, kam zařadit tento problém a z jakého úhlu pohledu ho řešit. Objasňuji zde často nesprávně používané pojmy klasifikace, lokalizace, detekce objektů, detekce textů a souvislosti mezi nimi. Dále se v této části nachází přehled použití hlubokých neuronových sítí pro analýzu obrazu. Zaměřuji se zde především na konvoluční neuronové sítě a jejich výhody proti jiným technikám.

Ve druhé části se věnuji popisu metod, které aplikují konvoluční sítě k řešení problémů detekce objektů. Nejdříve vysvětluji rozdíly mezi tzv. Single-stage a Two-stage detektory. Poté následuje rozbor konkrétních modelů spolu s jejich historickým vývojem. Tato sekce je zakončena porovnáním popsaných metod na obecném datasetu.

Ve třetí části nastíním v čem se liší detekce textu od detekce objektů. Popíši zde hlavní rozdíly a problémy specifické pro detekci textu. Dále zde popisuji princip fungování hlavních metod pro jejich řešení, včetně jejich souvislostí s modely popsanými v předchozí sekci.

V poslední části práce pak popisuji vlastní experimenty. Rozebírám zde použitou implementaci vybraných metod z předchozí kapitoly. Dále se zabývám postupem návrhu a generování vlastního datasetu pro učení. Na závěr představuji výsledky dosažené za použití různých metod, konfigurací a postupů učení.

2 Analýza problému a způsobu řešení

Chceme-li, aby byl počítač schopen zpracovat obrázek a získat z něj určité informace, je zřejmé, že se bude jednat o tzv. počítačové vidění. Jelikož konkrétně nás zajímá extrakce textu, nabízí se OCR - tedy optické rozpoznávání znaků. Nejstarší metody OCR spočívaly v prostém porovnávání znaků s předem definovanou množinou přípustných znaků pixel po pixelu. Moderní přístupy využívají tzv. feature extraction k redukci dimensionality problému a poté hledají nejpodobnější instance např. pomocí algoritmu k-nearest neighbors[1]. Pro klasifikaci textu je tedy OCR ideální. Pro naše účely však klasifikace nestačí, potřebujeme zapojit i lokalizaci. Na tento problém nejsou obvyklé OCR řešení připraveny.

Pokud zadání zobecníme a nebudeme se zabývat informací, že hledaný objekt je text, dostaneme se k problému detekce objektů v obrázku. Ten se skládá z lokalizace a klasifikace (Výjimka může být případ, kdy nás zajímá pouze jeden objekt na obrázku a jeho lokace. Pak se jedná o lokalizaci a klasifikaci, ale ne o detekci objektů. Takové aplikace však prakticky nemají využití a tak na ně v této práci nebude brán zřetel). Cílem je tedy najít objekt v obrázku a vrátit jeho třídu a lokaci. Zatímco klasifikace (přiřazení třídy dominantnímu objektu z obrázku) je obecně považována za vyřešenou, v oblasti modelů pro detekci objektu dochází k neustálému vývoji lepších postupů.

2.1 Hluboké učení versus OCR

Zatímco detekce a čtení textu v ohraničeném, kontrolovaném prostředí může být obvykle řešena pomocí heuristicky založených metod, u detekce textu v přirozených obrázcích (např. fotografie) čelíme problémům, které nejsme schopni pomocí těchto jednoduchých metod vyřešit. Jako příklady zmíněných problémů můžeme uvést:

- Šum způsobený snímačem - Šum z příručního fotoaparátu bude obvykle vyšší než u specializovaného skeneru. Dále musíme počítat s levnými, nekvalitními snímači.
- Úhly pohledu - V přirozeně zachycených snímcích se vyskytuje text nejrůzněji zakřivený a zdeformovaný v důsledku úhlu pořízení obrázku.
- Rozmazanost - Problém postihující převážně levné mobily se slabou, či zcela chybějící, stabilizací.
- Světelné podmínky - Nemůžeme předpokládat žádné podmínky – obrázek může být příliš tmavý, příliš světlý, mít přesvětlený bod v důsledku blesku fotoaparátu atd.
- Rozlišení - Obrázky ve velmi nízkém rozlišení představují výzvu pro detekci textu.
- Nerovná plocha - Text vytištěný na nerovné ploše (např. láhev) je stále jednoduše čitelný pro člověka, ale pro algoritmy je to značný problém.

Většina těchto problémů je spojena se zpracováním obrázků obecně. Účinnost hlubokého učení a především pak konvolučních neuronových sítí ve zpracování obrázků už byla prokázána v nejrůznějších publikacích[2][3][4], není tedy důvod předpokládat, že pro detekci textu nebude vhodná. Konvoluční neuronové sítě jsou také považovány za state-of-the-art technologii k řešení problému detekce objektů v obrázku [5][6][7].

2.2 Konvoluční neuronové sítě

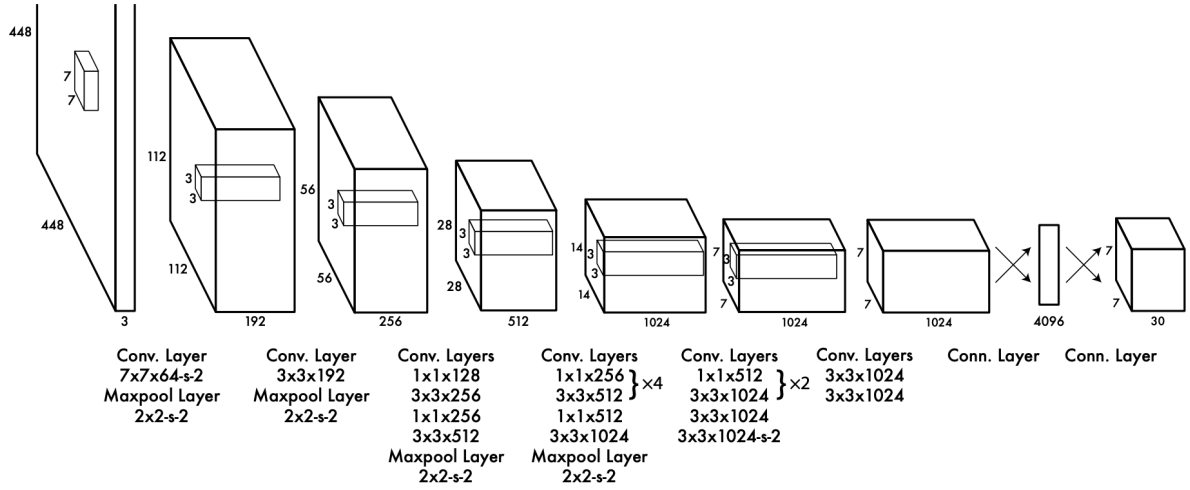
Konvoluční neuronová síť se skládá z konvolučních vrstev obvykle následovaných jednou, nebo více plně propojenými vrstvami. Pokud se v síti nenacházejí plně propojené vstvy, jedná se o tzv. plně konvoluční síť. Konvoluční vrstvu tvoří tzv. kernely (filtery) a zakončuje jí tzv. pooling vrstva.

2.2.1 Kernel

Velikost kernelu je dána hyperparametrem (obvykle se volí 1x1, 3x3, 5x5; na obrázku 1 5x5 v první vrstvě, dále 3x3). Třetí rozměr, tedy hloubka, je vždy roven hloubce vstupu. Každý kernel reprezentuje určitou feature (tím může být například svislá čára ve vstupním obrázku za předpokladu, že se kernel nachází v první konvoluční vrstvě a jeho vstupem je tedy originální obrázek). Kernel se poté aplikuje na všech možných pozicích ve vstupu. Na každé pozici jeho výstupní hodnota udává, jak velká je podobnost dané části obrázku k feature, kterou reprezentuje. Mezera mezi jednotlivými pozicemi kernelu (stride) je rovněž zadána hyperparametrem. Jako aktivační funkce kernelu se obvykle používa ReLU, tedy $f(x) = max(0,x)$. Jedna konvoluční vrstva běžně obsahuje několik kernelů (na obrázku 1 v první vrstvě 64, dále 128, v poslední vrstvě 256). Výstupem z kernelů je tzv. feature map, jejíž hloubka odpovídá počtu kernelů.

2.2.2 Pooling vrstva

Pooling vrstva se stará o redukci dimensionality a zároveň pomáhá bránit tzv. overfittingu, tedy ztrátě generalizace přílišným naučením na daném datasetu. Pooling vrstvu si stejně jako Kernel můžeme představit jako čtverec s velikostí zadanou hyperparametrem (na obrázku 1 2x2). Z tohoto čtverce se na výstup přenese obvykle největší hodnota (max-pooling), někdy také průměrná hodnota (avg-pooling). Tato operace se provede pro všechny kernely - hloubka vstupu se tedy poolingem nemění.



Obrázek 1: Příklad architektury konvoluční sítě [8]

V klasické plně propojené sítí je každému pixelu ze vstupu přiřazena váha pro každý neuron v první vrstvě. To způsobuje, že model pro komplexní obrázky roste do neúnosných rozměrů. Oproti tomu jsou neurony z konvoluční vrstvy napojeny pouze na malý počet vstupních pixelů (podle velikosti filtru) a váhy jsou sdíleny mezi všemi neurony vrstvy. Toto řídké propojení nám umožňuje přidávat více vrstev bez významného poklesu výkonu. Konvoluční vrstvy se v modelu chovají jako feature extractor - výrazně zmenšují dimensionalitu vstupu, ale zároveň zachovávají informace významné pro další zpracování. Na konci modelu pak zpravidla najdeme jednu, nebo více plně propojených vrstev, které klasifikují výstup z konvolučních vrstev.

Architektura konvolučních neuronových sítí nám umožňuje zpracování obrázku za použití pouze zlomku výpočetní síly, která by byla potřeba pro učení plně propojené sítě na stejný problém. Rozdíl v přesnosti je přitom zanedbatelný.

3 Detekce objektů

Detekce objektů je název zastřešující lokalizaci a klasifikaci objektu. V oblasti klasifikace se konvoluční neuronové sítě osvědčily jako velmi univerzální nástroj, pomocí kterého bylo možné, za podmínky použití dostatečného datasetu, dosáhnout více než uspokojivé přesnosti. Např. článek [9] z roku 2012 popisuje dosažení chyby 0,23% na datasetu MNIST. Síť AlexNet publikována téhož roku pak dosáhla do té doby nevídaného výsledku 15,3% top-5 error (metrika udavající procento případů, kdy ani jedna z nejlepších 5 odpovědí sítě nebyla správná) na podmnožině datasetu ImageNET obsahujícího tisíc tříd a celkem zhruba milion obrázků. Dá se říci, že rok 2012 představoval v oblasti konvolučních sítí průlom, a odstartoval jejich dominanci v oblasti počítačového vidění. Klasifikace však znamená pouze predikci třídy dominantního objektu na obrázku. Jakmile máme na jednom obrázku více objektů zájmu, narážíme na architektonický limit - výstupní vrstva má konstantní velikost, tudíž není možné klasifikovat současně více objektů. Problém může nastat také při potřebě klasifikovat malé objekty. Například vadné spoje na desce plošných spojů. Na první pohled se tato situace může zdát jako vhodná úloha pro klasifikaci, jelikož nás zajímá pouze třída celé desky (funkční, nefunkční). Charakteristiky umožňující toto rozdělení však mohou být příliš malé v poměru k celkové velikosti vstupního obrázku a síť je tak bude ignorovat.

Jako naivní přístup k řešení těchto problémů se nabízí vstupní obrázek rozdělit na několik regionů a tyto regiony následně použít jako vstup pro klasifikační model vždy jeden po druhém. Ze shromážděných výsledků bychom pak vybrali ty s největším skóre (finální vrstva v klasifikačních konvolučních modelech mívá zpravidla aktivační funkci softmax, výsledkem pak je vektor o velikosti počtu tříd, obsahující pro každou třídu pravděpodobnost, že daný objekt patří do této třídy). Zde je však potřeba si uvědomit, že objekty zájmu se mohou v obrázcích vyskytovat v libovolné velikosti. Pokud bychom například chtěli detekovat osoby na fotografiích, musíme počítat s velikostí regionu např. 30x100 pro stojící osobu, 150x45 pro ležící osobu blíže k fotoaparátu, 50x50 pro sedící osobu atd. Právě popsaný algoritmus se nazývá sliding window. Je jasné, že množství takových regionů, a tím pádem množství průchodu klasifikátorem pro jeden vstup, je neúnosné. Z toho vyplývá, že klíčová obtíž v detekci objektů spočívá v co možná nejoptimálnějším návrhu regionů pro klasifikaci. V současné době už existuje mnoho metod řešících zmíněné problémy, ty nejvýznamnější z nich jsou spolu s jejich vývojem popsány v následujících sekcích.

3.1 Single-stage a multi-stage detektory

State-of-the-art modely pro detekci objektů se dají rozdělit podle architektonického přístupu ke spojení klasifikace a lokalizace do dvou skupin: single-stage a two-stage detektor.

První modely využívající k řešení detekci objektů konvoluční neuronové sítě (např. R-CNN[10], Fast R-CNN[11]) se skládaly ze dvou částí. První část tvořil algoritmus, který měl za úkol navrhovat v obrázku boxy (výřezy), které by mohly obsahovat objekt. Ve zmíněných modelech

se o tuto část staral algoritmus Selective Search[12], založený na principu segmentace, pozdější modely už i k návrhu boxů využívají konvoluční sítě. Každý z navrhnutých boxů se poté v druhé části použil jako vstup do konvoluční sítě, která měla za úkol klasifikaci objektu uvnitř.

Tento postup tedy používají two-stage modely, zjednodušeně jde o to, že lokalizace a klasifikace probíhá zvlášť ve dvou krocích. V této kategorii v současné době s ohledem na nejlepší dosažené výsledky vede model Light-Head R-CNN[13].

V době, kdy two-stage model Faster R-CNN[11] dominoval v oblasti přesnosti, výpočetní náročnost byla příliš vysoká pro použití v reálném čase, například pro detekci objektů ve videu. Rychlost se měřila spíše v sekundách na obrázek, místo obvyklého počtu zpracovaných obrázků za sekundu (FPS). Tento nedostatek někteří přisuzovali právě rozdělení problému na dva kroky, a tak se začaly vyvíjet single-stage modely, které oba problémy spojily do jednoho kroku.

První publikovaný single-stage model YOLO[7] se chlubí několikanásobným překonáním rychlosti Faster R-CNN (7 FPS Faster R-CNN, 45 FPS YOLO) při současném snížení mAP o 10 bodů. Jedná se o porovnání na datasetu Pascal VOC 2007. Zároveň však autoři zdůrazňují, že YOLO pokulhává u detekce malých objektů, zatímco u velkých objektů je přesnost téměř totožná s Faster R-CNN. Tohoto zrychlení se podařilo dosáhnout díky výraznému zjednodušení modelu – ten u YOLO spočívá pouze v jedné konvoluční síti. Tento model tedy jednoznačně ukázal, že má cenu se one-stage detektory zabývat a brzy následovaly další (SSD, YOLOv2, YOLOv3).

Zatím se nepodařilo prokázat obecnou převahu jedné ze zmíněných skupin. Vývoj tak stále probíhá a inovace vznikají na obou stranách. Na vrcholu pomyslného žebříčku výkonnosti se zástupci obou skupin střídají víceméně pravidelně.

3.2 Evaluace modelu

Pro hodnocení přesnosti modelů pro detekci objektů se používá již zmíněná metrika mAP (mean Average Precision). Ta představuje průměr maximálních přesností pří různých hodnotách recall. Recall hodnota uvádí kolik procent všech správných výsledků model označil jako správné, bez ohledu na počet nesprávných označení. Abychom byli schopní definovat mAP, musíme nejdříve zavést následující pojmy. Ty se vztahují ke každému vyhodnocovanému obrázku:

- Intersection over Union (IoU): Poměr průnik:sjednocení mezi predikovaným boxem a odpovídajícím ground truth boxem
- Confidence score: znázorňuje pravděpodobnost, že predikovaný box obsahuje určený objekt. Confidence score = IoU.
- True positives (TP): počet objektů, které model odhadl správně (vrátil pro ně box, který má větší IoU, než stanovená prahová hodnota, nejčastěji 0.5)
- False positives (FP): počet boxů, které model určil jako nějaký objekt, ale IoU je pro všechny instance daného objektu nižší než prahová hodnota

- False negatives (FN): počet objektů, pro které model nevrátil box
- Precision: TP/(TP+FP), jak velká část označených objektů, je označeno správně. Počítá se pro každou třídu.
- Recall: TP/(TP+FN), jak velká část skutečných objektů je označeno. Opět pro každou třídu.

Ukázka výstupu modelu pro detekci objektů může vypadat například takto:

Tabulka 1: Příklad výstupu detekčního modelu pro jednu třídu

TP/FP	Precision	Recall
TP	1/1	1/3
FP	1/2	1/3
TP	2/2	2/3
FP	2/3	2/3
FP	2/4	2/3
TP	3/5	3/3
FP	3/6	3/3

Každý řádek tabulky [1] zde představuje jednu predikci modelu, přičemž řádky jsou seřazeny sestupně podle *confidence score*. První řádek se dá popsat následovně. První odhad modelu je správný (TP). Přesnost se rovná jedné (provedli jsme jeden odhad a z toho byl jeden správný). Recall se rovná jedné třetině (nalezli jsme jeden objekt z celkových tří).

Nyní můžeme pro každou třídu spočítat průměr nejvyšších přesností pro každou hodnotu *recall*. Tato metrika se nazývá Average Precison (AP). Výpočet provádíme pro jedenáct hodnot *recall* daných intervalem <0;1> a krokem 0,1 podle následujícího vzorce:

$$AP = \frac{1}{11} * \sum_{r \in \{0;0,1;...;1\}} max_{\hat{r} \geq r} p(\hat{r}),$$

kde r označuje jednotlivé hodnoty *recall* a $max_{\hat{r} \geq r} p(\hat{r})$ představuje nejvyšší dosaženou hodnotu *precision* pro danou hodnotu *recall*.

Metrika mAP pak znamená jen zprůměrování hodnot AP přes všechny třídy. Výsledná hodnota mAP nám tedy říká, jak přesně model funguje na daném datasetu. V některých publikacích se označení mAP a AP zaměňují, v takovém případě by z kontextu mělo vyplývat, zda se jedná o výpočet pro konkrétní třídu, nebo průměr přes všechny třídy.

Další metrika využívána pro detektory objektů především v kontextu s detekcí textu je F-measure (také F-score, F1 score). Ta se opět vypočítává z hodnot Precision a Recall podle následujícího vzorce:

$$F = 2 * \frac{precision * recall}{precision + recall}$$

V publikacích jednotlivých modelů obvykle nalezneme tabulku obsahující jejich přesnost v jednotce mAP na určitém datasetu a porovnání s jinými modely. Obecně je velmi těžké porovnávat mezi sebou modely pro detekci objektů, jelikož pro korektní porovnání musíme znát mAP hodnoty na stejném datasetu. Dále vstupují do hry volitelné parametry (například IoU rozhodující, zda je daný box TP, nebo FP). Nezanedbatelný vliv na výsledky mají také různé postupy učení (použití předučeného modelu, různé formy předzpracování datasetu, apod.).

3.3 Two-stage modely

Vývoj modelů pro detekci objektů založených na použití konvolučních neuronových sítí, osvědčených z oblasti klasifikace, odstartoval model R-CNN. Záhy poté titéž autoři přišli se zásadně vylepšenou verzí Fast R-CNN. State-of-the-art v oblasti detekce momentálně představuje třetí evoluce původní R-CNN - Faster R-CNN. Tyto tři modely způsobily zásadní průlom v detekci objektů a z toho důvodu jim věnuji několik následujících sekcí.

3.3.1 R-CNN

Model R-CNN [10] byl publikován v roce 2014. Byl to první model, který ukázal, že použití konvoluční sítě může vést k významnému zvýšení přesnosti na datasetech pro detekci objektu. Konkrétně se tomuto modelu podařilo dosáhnout přesnosti 53,3% na datasetu PASCAL VOC 2012[14], což je zlepšení o více než 30% proti předchozímu nejlepšímu výsledku. Podobné úspěchy zaznamenal také na datasetech PASCAL VOC 2007 a PASCAL VOC 2010.

Fungování R-CNN lze shrnout do následujících tří kroků:

1. Vygenerování návrhů regionů, které pravděpodobně ohraničují nějaký objekt zájmu
2. Feature extraction pro každý navržený region
3. Klasifikace výstupu výsledné feature map

Selective Search

Pro generování návrhů regionů je zde použit algoritmus Selective search[12]. Cílem tohoto algoritmu je v co nejkratším čase vygenerovat regiony s velmi vysokou hodnotou recall. Je založen na hierarchickém seskupování podobných regionů. Podobnost je zde vyhodnocována na základě barvy, typu povrchu, velikosti a podobnosti tvarů.

Nejdříve se provede segmentace obrázku pomocí metody [15]. Výsledný segmentovaný vstup je zobrazen na obrázku 2. Samotné segmenty jako regiony zájmu nestačí, protože většina objektů je spojena z více regionů a naopak některé objekty nejsou kompletně reprezentovány jedním segmentem.

Obrázek 2: Příklad segmentace obrázku. Vlevo vstup, Vpravo výstup.[16]

Segmenty z předchozího kroku se tedy přidají do množiny návrhů regionů, ale poté se pokračuje spojením přilehlých regionů na základě podobnosti. Touto operací vzniká nová množina segmentů, tentokrát o větších rozměrech. Nově vzniklé segmenty jsou opět přidány do množiny návrhů regionů a celý proces se iterativně opakuje až do doby, kdy se všechny segmenty spojí v jeden. Každému segmentu je přiřazen tzv. rank. Rank se přiřazuje v opačném pořadí, než v jakém byly regiony přidány. To prakticky znamená, že čím větší segment, tím menší rank je mu přiřazen. Poslední segment, tedy ten, který obsahuje celý vstupní obrázek, je ohodnocen rankem 1. Rank bude dále využit v následujících krocích k určení pořadí výsledných regionů.

Celý popsaný proces se v rámci algoritmu opakuje několikrát, pro různé strategie. Strategie se liší v barevném pásmu vstupu, v použitých metrikách podobnosti, nebo různou počáteční množinou segmentů (vlivem volitelných parametrů segmentační metody).

Výsledkem algoritmu selective search je tedy sjednocení všech množin návrhů regionů z různých strategií, seřazena podle metriky jejich ranků. Výsledný rank regionu získáme podle následujícího vzorce:

$$r = \sum RND * i_j$$

kde j je index strategie, i_j je rank daného regionu v j-té strategii a RND je náhodně vygenerované číslo z intervalu [0,1].

Pořadí těchto ranků znázorňuje tzv. objectness metriku. Objectness udává pravděpodobnost, že daný region obsahuje nějaký objekt zájmu. Základní myšlenka tohoto hodnocení spočívá v předpokladu, že čím větší region, tím větší objectness. To samozřejmě neplatí vždy, proto je ve vzorci rank regionu přenásoben náhodným číslem, tedy náhodně snížen. Zvýšení šance menších regionů, které skutečně obsahují objekt zájmu, na získání menšího ranku je dále zajištěna duplicitou těchto regionů v několika strategiích (pokud region skutečně obsahuje objekt zájmu, je pravděpodobné, že se dobře umístí ve více strategiích a to znamená větší šanci na náhodné snížení ranku). Nakonec se všechny duplicitní regiony odstraní, zůstane pouze ten s nejnižším rankem.

Jednou z hlavních výhod tohoto algoritmu je fakt, že si můžeme zvolit kolik nejlepších regionů chceme vygenerovat. To v praxi znamená, že rozhodnutí kvantita versus kvalita je čistě na nás. Pro účely R-CNN bylo použito 2000 nejlepších návrhů.

Feature extraction

O feature extraction se v R-CNN stará konvoluční síť AlexNet[17]. Ta obsahuje pět konvolučních vrstev a 2 plně propojené vrstvy. Vstupem do této sítě jsou návrhy reginů z předchozího kroku. Architektura konvoluční sítě (přesněji plně propojené vrstvy na jejím konci) neumožňuje zpracovávání proměnné velikosti vstupů, tudíž nejprve musíme upravit velikost regionů. K upravení autoři R-CNN použili obyčejnou metodu stretch, tedy roztažení obrázku do požadovaného rozměru. Síť AlexNet je připravena na velikost vstupu 227x227x3. Výstupem pak je vektor o 4096 dimenzích.

Klasifikace

Klasifikaci zajišťují Support Vector Machine (SVM)[18] klasifikátory. Pro každou třídu se zde trénuje SVM, který přijímá na vstupu vektor z předchozího kroku a vrací pravděpodobnost, že daný region obsahuje objekt odpovídající třídy. Po klasifikaci je pro každý návrh regionu, jehož IoU s některým z ground truth boxů je větší, než 0,6, se v učící fázi navíc provádí regrese souřadnic regionu pomocí bounding-box regresoru. Tímto způsobem se model učí zpřesňovat odhad boxů ohraničujících objekty.

Přestože model R-CNN představoval významný přínos v dané problematice, stále s ním byly svázány dva významné nedostatky:

- Nutnost zpracování 2000 regionů jeden po druhém, je stále výpočetně velmi náročné jak pro trénování, tak pro následná aplikace. Použití tohoto modelu pro detekci objektu v reálném čase je stále nemyslitelné. Kromě výpočetní složitosti zde narážíme i na tu prostorovou. V průběhu trénování je třeba krátkodobě uchovávat až stovky GB dat.
- Systém zajišťující generování návrhů regionů (algoritmus Selective search) se nedá trénovat, to může vést ke generování špatných návrhů.

Autoři R-CNN na závěr zdůrazňují přínos předtrénování modelu na datasetu pro klasifikaci, a následné doučení modelu na datasetu pro detekci objektu. Při tomto postupu (předtrénování na datasetu ILSVRC) uvádějí zvýšení hodnoty mAP na datasetu PASCAL VOC z 46% na 54%. Vzhledem ke složitosti výroby datasetu pro detekci objektu a tudíž obecnému nedostatku použitelných dat oproti datasetu klasifikačnímu, byl tento důkaz zásadní pro další vývoj modelů pro detekci objektu.

3.3.2 Fast R-CNN

Tento model byl publikován v roce 2015[11] původními autory R-CNN. Vychází ze svého předchůdce a adresuje hned několik jeho nedostatků. Velmi hlubokou síť VGG16[19] dokázal natrénovat na dataset PASCAL VOC 9x rychleji, a při predikci byl rychlejší 213x oproti R-CNN. Autoři v této publikaci také rozebírají vliv množství návrhů regionů na celkovou přesnost sítě. Mezi nejvýznamnější vylepšení modelu patří:

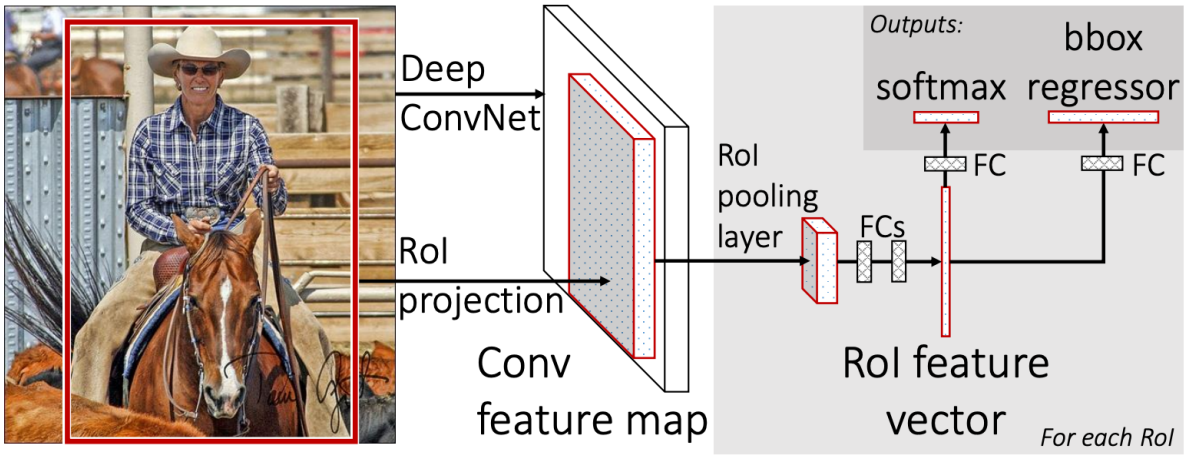
- Sdílení výpočtů konvolučních vrstev - feature map
- Možnost trénovat jak klasifikaci, tak regresi boxů zároveň
- Možnost trénovat všechny vrstvy sítě
- Minimalizovaná prostorová náročnost - není nadále potřeba ukládat data na disk

Kombinace výše zmíněných faktorů má přirozeně za následek celkové snížení náročnosti, a zároveň zvýšení přesnosti.

Architektura

Vstupem do Fast R-CNN je stejně jako u R-CNN obrázek a množina návrhu regionů (opět se používá algoritmus selective search). Rozdíl však spočívá v pořadí operací. Fast R-CNN nejprve zpracuje obrázek jako celek konvoluční sítí. Následné operace už probíhají opět jednotlivě pro každý navrhnutý region. Z výsledné feature map je pomocí RoI (Region of Interest) pooling vrstvy extrahován tzv. feature vektor pevné délky, jehož pozice odpovídá pozici navrhnutého regionu z původního obrázku. Schéma architektury je zobrazeno na obrázku (3). Tento vektor pak postupuje dále do dvou plně propojených vrstev, jejíž výsledek se zpracuje jak softmax vrstvou (na rozdíl od SVM v R-CNN) pro klasifikaci, tak regresorem pro upřesnění lokace regionu.

Zásadní změna architektury tedy spočívá ve sdílení výpočtu konvolučních vrstev, který se nyní provádí pouze jednou pro každý obrázek, namísto jednou pro každý návrh regionu jak tomu bylo u R-CNN. Z toho vyplývá, že čím více konvolučních vrstev používáme, tím výraznější rozdíl v rychlosti mezi těmito modely pozorujeme. Další významná vlastnost této architektury je možnost trénovat konvoluční vrstvy jako celek pomocí zpětného šíření chyby. To bylo samozřejmě v předchozím případě, kdy každý návrh regionu měl vlastní váhy konvolučních vrstev značně komplikované.



Obrázek 3: Schéma modelu Fast R-CNN. Červený obdélník značí návrh regionu. Následuje jeho projekce na feature map, jejíž výsledkem je feature vektor. Ten se dále zpracovává přes plně propojené vrstvy až do rozdvojení mezi klasifikátor a regresor. Celá pravá strana obrázku (oblast s šedým pozadím) je aplikována pro každý návrh regionu. [11]

RoI pooling vrstva

Dosavadní konvoluční neuronové sítě byly schopny zpracovávat pouze obrázky pevné velikosti. Toto omezení vyplývá z vlastností plně propojených vrstev - při konstrukci takové vrstvy musíme vědět, s jakou velikostí vstupu bude pracovat. Jelikož v případě detekce objektů není praktické být omezován pevnou velikostí obrázku, obvyklé metody spoléhaly na předzpracování před vstupem do plně propojené vrstvy, tzn. každý obrázek byl upraven na potřebnou velikost. Tato praktiky je problémová, protože u obrázků může dojít ke zkreslení (v případě roztažení), či dokonce ztrátě (v případě oříznutí) informací.

RoI pooling vrstva přichází s jiným přístupem, než standardní pooling vrstvy. Nedefinujeme hyperparametrem velikost jejího okna, nýbrž definujeme požadovanou velikost výstupu. Vstupem do této vrstvy je feature map z konvolučních vrstev a množina návrhů regionů. Velikost okna je pak dána dvojicí $(w/W, h/H)$ kde w,h reprezentují šířku a výšku vstupu - v našem případě tedy regionu a W,H je požadovaná šířka a výška výstupu. Pro samotný pooling je pak použita klasická metoda max pooling. RoI pooling vrstva adaptuje myšlenku Spatial Pyramid Pooling popsanou v článku [20].

Kombinovaná chybová funkce (Multi-task loss)

Model Fast R-CNN produkuje současně dva výstupy. První výstup představuje rozdělení pravděpodobnosti $p = (p_0, ..., p_C)$ pro C+1 tříd (jedna třída navíc definuje pozadí). Toto rozdělení představuje výstup softmax funkce, jejíž vstupem je tradičně výstup z plně propojených vrstev. Druhý výstup pochází z regresoru a představuje posun oproti původnímu boxu

$t^c = (t_x^c, t_y^c, t_w^c, t_h^c)$, kde c je index třídy a x,y,w,h jsou indexy souřadnic boxu. Každý region je označen ground-truth třídou u a ground-truth boxem v.

Chyba klasifikace, je pak určena funkcí

$$L_{cls}(p, u) = -log(p_u)$$

Chybová funkce regresoru

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L1}(t_i^u - v_i)$$

kde

$$smooth_{L1}(x) = \begin{cases} 0,5x^2 & \text{pokud } |x| < 1 \\ |x| - 0,5 & \text{jinak} \end{cases}$$

je robustní L1 funkce používaná v regresních metodách, která je méně náchylná na odlehlá pozorování, než L2 funkce používaná v R-CNN. Při použití L2 funkce s neohraničenými vstupy může dojít k tzv. exploding gradient fenoménu, tedy k náhlým velkým změnám vah v důsledku nakumulování vysokých hodnot chyb.

Celková kombinovaná chyba modelu je tedy definována jako

$$L(p, u, t^u, v) = L_{cls} + \lambda[u \geq 1]L_{loc}(t^u, v)$$

kde λ je hyperparametr určující poměr významnosti mezi jednotlivými chybovými funkcemi. $[u \geq 1]$ pak znázorňuje funkci vracející 1, pokud je u větší než 1 a 0 v opačném případě. Podle konvencí $u = 0$ značí třídu pozadí, a v tom případě nás při výpočtu chyby nezajímá pozice boxu.

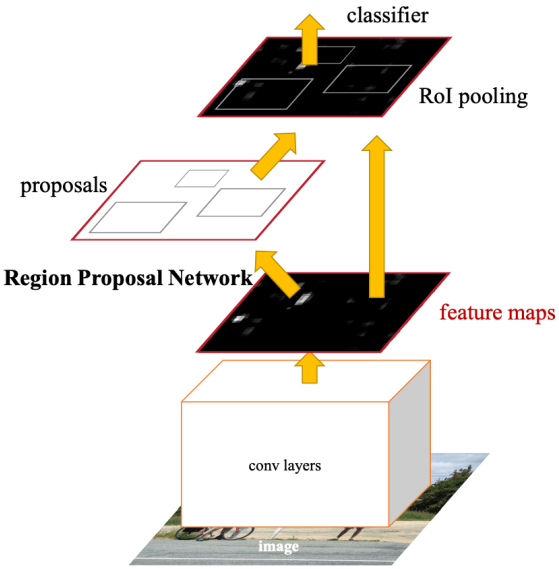
Minimalizace této funkce kombinované chyby nám umožňuje trénovat klasifikaci i lokalizaci pomocí zpětného šíření chyby zároveň, což je klíčová vlastnost Fast R-CNN.

Fast R-CNN model přinesl významné snížení výpočetní a prostorové náročnosti, čímž zminimalizoval jeden z velkých nedostatků R-CNN. Čas predikce se díky tomu snížil ze zhruba 49 sec/obr na 2,3 sec/obr. Další z problémů R-CNN však stále přetrvává - návrh regionů ke klasifikaci stále probíhá mimo model fixním algoritmem, který není možné trénovat. Několikanásobné zrychlení procesu predikce zároveň ukázalo, že naprostou většinu času nyní zabírá právě generování návrhů regionů.

3.3.3 Faster R-CNN

Model Faster R-CNN[5] představuje další krok v evoluci metod založených na samostatném návrhu regionů a momentálně se jedná o state-of-the-art řešení problému detekce objektů z hlediska přesnosti. Byl publikován v roce 2015 a opět se zaměřuje na eliminaci největších problémů předchozího modelu. Těmi jsou jednoznačně časová náročnost algoritmu selective search a nemožnost jeho trénování. Velká výhoda použití hlubokých neuronových sítí pro detekci objektu je mož-

nost masivní paralelizace jejich výpočtů - tudíž možnost využít potenciál GPU. Tuto možnost segmentační algoritmy postrádají. I kdyby se však povedlo nějakou vhodnou metodu naimplementovat s podporou GPU výpočtů, stále by se jednalo o samostatný proces nesouvisející se samotnou konvoluční sítí. To autoři Faster R-CNN považovali za nevyužití potenciálu a hledali způsob, jak použít pro návrh regionů výpočty konvolučních vrstev, které už máme k dispozici. Na této myšlence tak založili svou metodu pojmenovanou Region Proposal Network (RPN). Výsledná architektura Faster R-CNN je téměř totožná s Fast R-CNN, jediný rozdíl spočívá ve způsobu návrhu regionů. Architektura Faster R-CNN je znázorněna na obrázku (4).



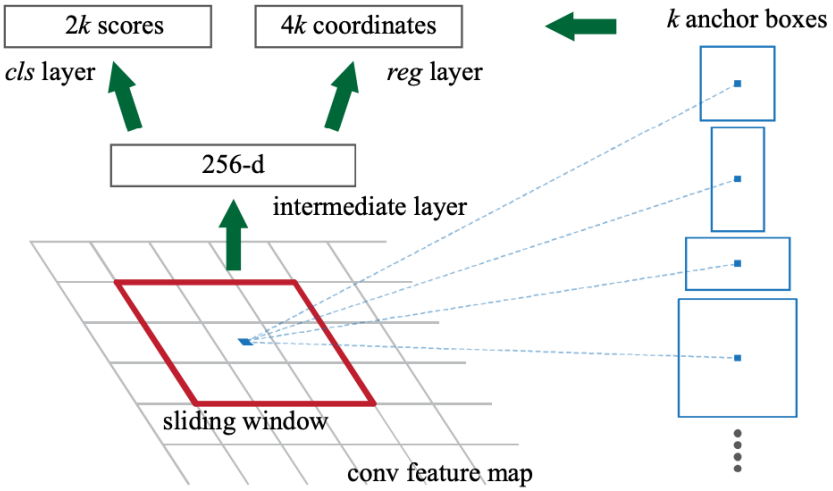
Obrázek 4: Architektura Faster R-CNN. Obrázek zachycuje použití feature mapy jak sítí RPN (červeně ohraničená síť vpravo), tak klasifikátorem. Díky sjednocení do jedné sítě je také možné trénovat celý model standardním algoritmem pro zpětné šíření chyby. [5]

Region Proposal Network (RPN)

RPN je síť, která přijímá na vstupu obrázek libovolné velikosti, a produkuje na výstup množinu boxů spolu s jejich odpovídajícím objectness skóre. Jelikož cílem Faster R-CNN je sdílet konvoluční vrstvy s modelem pro detekci objektu, odpovídá architektura RPN konvoluční síti použiti použité pro detekci objektu (autoři opět zůstali u sítí AlexNet a VGG).

RPN operuje nad feature map, tedy výstupem z konvolučních vrstev sítě, na principu posuvného okénka o rozměru n x n (v článku $n = 3$). Na každé pozici okénka je pak vygenerováno k regionů. Parametr k zde znamená počet rozměrů boxů, zadaných hyperparametry. Rozměry se zde zadávají ve formě šířek a poměrů stran. Například v článku je používaná hodnota $k = 9$. Konkrétní použité rozměry jsou 128, 256 a 512, a pro každý rozměr nás zajímají 3 poměry stran 1:1, 1:2, 2:1. Každá pozice posuvného okénka je ve formě vektoru fixní délky předána plně propojeným vrstvám pro klasifikaci a regresi boxů. Klasifikační vrstva je zde reprezentována pouze

dvoucestnou softmax vrstvou (obsahuje objekt/neobsahuje objekt) a velikost jejího výstupu je tedy $2k$. Regresní vrstva vrací výstup o velikosti $4k$ udávající pozici boxu relativní ke středu okénka.



Obrázek 5: Princip RPN. Z každé pozice posuvného okénka (červené ohraničení) se extrahuje vektor fixní délky (např. 256). Z tohoto vektoru se poté produkuje skóre a souřadnice regionu pro každý nastavný rozměr boxů. [5]

Učení Faster R-CNN

Učení Faster R-CNN probíhá standardním způsobem s využitím zpětného šíření chyby. K tomu využíváme podobně jako u Fast R-CNN kombinovanou chybovou funkci. Tentokrát se však proces učení komplikuje, protože učíme dvě samostatné sítě, které však sdílí váhy konvolučních vrstev. Obě tyto sítě přitom produkují vlastní výstupy pro klasifikaci a regresi a tím pádem i vlastní chybu. Učení tedy probíhá v následujících čtyřech krocích.

- Učení samotné RPN.[1]
- Učení samotného detektoru Fast R-CNN.[2]
- Učení celého modelu Faster R-CNN, váhy společných konvolučních vrstev jsou však fixovány, upravujeme pouze váhy samostatných vrstev RPN.
- Učení celého modelu Faster R-CNN, váhy společných vrstev zůstávají fixovány, upravujeme pouze váhy samostatných vrstev detektoru. Fast R-CNN.

[2] Jako model pro RPN a Fast R-CNN musíme použít stejnou síť (např. AlexNet, VGG, apod.) abychom umožnili následné sdílení konvolučních vrstev. Doporučuje se použít síť předučenou na některém z obecných datasetů (např. PASCAL VOC).

Učení a predikce RPN

Jelikož v typickém vstupním obrázku zpravidla zabírají objekty zájmu výrazně menší plochu než pozadí, také naprostá většina regionů vygenerovaných RPN bude představovat negativní vzorky. Abychom vyvážili trénovací vstupní data, použijeme pro učení pouze 256 náhodně vybraných regionů tak, aby vzorků negativní:pozitivní byl 1:1. Neméně důležité je také způsob zpracovávání regionů, jenž zasahují mimo hranice obrázku. Tyto regiony ve fázi učení ignorujeme, jinak by způsobovaly vysokou chybu a tím značně ztížily konvergenci. V typickém vstupním obrázku o velikosti 1000x600 dostaneme zhruba 20000 regionů, z čehož nám po odstranění těch přesahujících zůstane okolo 6000. Ve fázi predikce pak přesahující regiony bereme v úvahu s tím, že před výstupem je ořízneme podle hranic vstupního obrázku.

Ze zbývajících 6000 regionů se stále značná část prolíná. Abychom snížili redundanci používáme metodu non-maximum suppression (NMS). Ta se dá shrnout ve třech krocích. Nejprve se odstraní všechny regiony, jejichž objectness skóre je menší než 0,7. Následně se zvolí region s nejvyšším objectness skóre jako referenční. Nakonec se odstraní všechny regiony, jejichž IoU s referenčním regionem je větší než 0,5. Po tomto procesu obvykle zůstane cca 2000 regionů. Počet nejlepších boxů, který Faster R-CNN ve výsledku vrátí se definuje hyperparametrem. Autoři používají pro učení hodnotu 2000, pro predikce pak hodnotu nižší (např. 300).

3.4 Single-stage modely

Přestože představují two-stage modely velmi silný nástroj, který dosahuje špičkových výsledků v oblasti přesnosti detekce objektů, jejich robustní přístup má za následek vyšší výpočetní, a pochopitelně i časovou náročnost. Zde se tedy otvírá cesta pro nové přístupy. Jedním takovým jsou i single-stage detektory. Jejich hlavní prioritou je rychlost, a v podstatě až jejich příchod nám umožnil použití detekce objektů v reálném čase.

3.4.1 You Only Look Once (YOLO)

Prakticky současně s modelem Faster R-CNN se objevila publikace You Only Look Once[7] řešící stejný problém novým přístupem. Všechny dosavadní metody fungovaly na společném principu - nejprve extrahovaly regiony, které by mohly obsahovat nějaký objekt zájmu a na tyto regiony následně aplikovaly klasifikátor. Samotný detektor tak nikdy nerozhodoval na základě celého obrázku, ale pouze na základě jeho výřezu. Naproti tomu YOLO pracuje s detekcí objektů jako s regresním problémem, jehož cílem je predikovat souřadnice objektů společně s pravděpodobností příslušnosti daného objektu do každé třídy. Celý tento proces zajišťuje pouze jedna konvoluční síť a tudíž jeden učící proces odlaďuje současně jak klasifikaci tak lokalizaci. Tímto přístupem se model YOLO stal prvním, který bylo možné použít pro opravdovou detekci v reálném čase (45 FPS; 155 pro odlehčený model Fast YOLO, zde však už znatelně klesá přesnost). Díky zpracovávání celého obrázku najednou YOLO bere v úvahu také kontext. To pak vede k nižšímu

(až polovičnímu) počtu falešných detekcí oproti single-stage modelům. Na druhou stranu tento přístup přináší snížení přesnosti především na objektech velmi malých rozměrů.

Základní princip

Vstupní obrázek je rozdělen do mřížky SxS. Buňka mřížky je zodpovědná za detekci objektu, jestliže obsahuje střed daného objektu.

Každá buňka pak predikuje B boxů a jejich confidence skóre. Confidence skóre značí jak pravděpodobné je, že daný box obsahuje objekt, a také jak přesně je daný objekt označen tímto boxem. Vzorec pro výpočet skóre pak vypadá následovně:

$$conf = Pr(Object) * IoU_{pred}^{truth},$$

kde *Pr(Object)* je pravděpodobnost, že daný box obsahuje *Object*, IoU_{pred}^{truth} značí jak moc se překrývají predikovaný a ground truth box.

Predikce každého boxu je definována pěti hodnotami *x,y,w,h,conf*. Koordináty (x,y) znázorňují relativní pozici středu boxu vzhledem k hranicím buňky. Hodnoty w,h označují relativní šířku a výšku vůči celému obrázku.

Každá buňka pak predikuje C pravděpodobností příslušnosti objektu, jež buňka obsahuje do jednotlivých tříd $Pr(Class_i|Object)$. Pravděpodobnost je podmíněna tím, že buňka obsahuje objekt. Nezávisle na počtu boxů, buňka predikuje vždy pouze jednu monžinu pravděpodobností.

Při predikci se pak pro každý box přenásobí jeho confidence skóre s pravděpodobnostmi výskytu jednotlivých tříd, čímž dostaneme pro každou dvojici (*třída*, *box*) celkové skóre znázorňující jak pravděpodobnost výskytu objektu dané třídy v daném boxu, tak míru přesnosti daného boxu.

Architektura

Při modelování sítě se autorři inspirovali u klasifikační sítě GoogLeNet[21]. Základ tvoří 24 konvolučních vrstev následovaných 2 plně propojenými vrstvami. Velikost výstupního tensoru sítě se rovná SxSxC. Konvoluční filtry střídají rozměry 1x1 a 3x3, max-pooling vrstvy mají velikost 2x2 (stride 2).

Chybová funkce

Při učení minimalizujeme následující kombinovanou chybovou funkci:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x})^2 + (y_i - \hat{y})^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}})^2]$$

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C}_i)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{noobj}(C_i-\hat{C}_i)^2+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2$$

kde $\mathbb{1}_i^{obj}$ značí zda se objekt *obj* nachází v buňce i a $\mathbb{1}_{ij}^{obj}$ značí, zda box j v buňce i je zodpovědný za predikování objektu *obj*. Parametry λ_{coord} a λ_{noobj} pak slouží k nastavení rozložení váhy mezi chybou souřadnic boxů a chybou confidence skóre boxů neobsahujících objekt. Obvykle se totiž v obrázcích nachází mnohem více buněk neobsahující žádný objekt, než těch, které nějaký objekt obsahují. Confidence skóre těchto buněk se tak blíží nule, což přenáší příliš velkou váhu na chybu buněk obsahující objekty a tím se celý model destabilizuje.

Tato chybová funkce je tedy navržena tak, aby penalizovala klasifikační chybu buňky pouze v případě, že se v dané buňce nachází nějaký objekt. Zároveň chyba souřadnic boxu se penalizuje pouze tehdy, je-li box zodpovědný za daný objekt (tzn. má největší hodnotu IoU vůči ground truth boxu daného objektu).

3.4.2 Single shot Multibox detector (SSD)

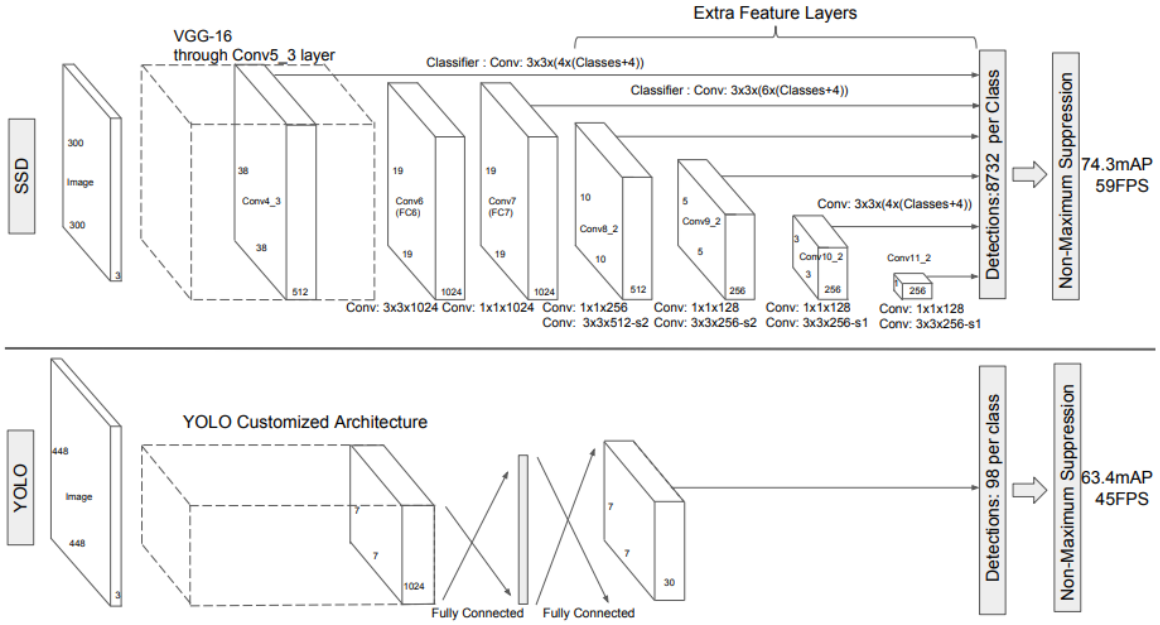
Na úspěchy modelu YOLO navazuje publikace z roku 2016 Single shot Multibox detector[6]. Přináší s sebou další zrychlení (59 FPS, vs. 45 FPS pro YOLO), nižší citlivost na vstupní rozlišení a především přesnost konkurující state-of-the-art modelu Faster R-CNN. Na zvýšení přesnosti má zásluhu zejména inspirace metodou MultiBox[22] a její modernizovanou formou MSC-MultiBox[23]. Princip těchto metod je podobný, jako např. RPN - z výstupu poslední konvoluční vrstvy se extrahuje pomocí posuvného okénka vektor, na který se poté uplatňuje klasifikace a regrese. Rozdíl je v tom, že MultiBox metoda se nezabývá pouze poslední konvoluční vrstvou, ale pracuje i s feature map vrstev předchozích. Tímto způsobem dokážeme pokrýt více velikostí objektů, jelikož informace o malých objektech se mohou během průchodu konvolučními vrstvami vlivem redukce dimenzionality ztrácet.

Princip

Celková architektura, i princip fungování je velmi podobný modelu YOLO. Základ SSD tvoří konvoluční síť VGG-16[19]. Plně propojené vrstvy VGG byly zaměněny za sadu postupně se zmenšujících konvolučních vrstev (variace MultiBox modelu). Každá tato vrstva je schopná predikce na základě aplikace výchozích boxů na svou feature map. Z toho tedy vyplývá, že model pro predikci vypadá pro každou vrstvu jinak. Na závěrečné filtrování predikcí je použita metoda non-maximum suppression (NMS), stejně jako ve Faster R-CNN.

MultiBox

MultiBox[22] je regresní metoda, jejíž cílem je vygenerovat návrhy regionů nezávisle na třídě objektů, kterou obsahují. Je založená na konvoluční síti s chybovou funkcí opět kombinující chybu



Obrázek 6: Porovnání architektury YOLO a SSD. Obrázek znázorňuje přístup inspirovaný principem Residual Networks[24]. Vybrané konvoluční vrstvy jsou napojenny nejen na následující vrstvy, ale zároveň na poslední vrstvu, která má na starost regresi. [6]

lokalizace a confidence skóre. Pro určení váhy těchto dvou chyb slouží opět hyperparametr, jímž se násobí chyba lokalizace. Myšlenka této metody je velmi podobná síti RPN a modelu YOLO. Také jsou zde aplikovány výchozí boxy předem daných rozměrů na výstupy konvolučních vrstev - feature map. Přínos metody MultiBox spočívá v tom, že se tyto boxy aplikují současně na více feature map různých velikostí. V praxi to vypadá tak, že vrstvy obsahující feature mapy pro detekci slouží jako vstup ne jen následujícím konvolučním vrstvám, ale také poslední vrstvě v síti (detektoru). Taková síť se nazývá Residual Network[24]. Díky tomu je celý proces detekce méně závislý na velikosti hledaného objektu.

Změny SSD oproti modelu MultiBox

V modelu SSD jsou výchozí boxy různých rozměrů a poměrů stran nastaveny manuálně (hyperparametry). Naproti tomu MultiBox pracoval s výchozími boxy vybranými tak, aby jejich IoU s libovolným ground-truth boxem bylo větší, než 0,5. Vyžadoval tedy předzpracování trénovacího datasetu ve formě generování těchto boxů. Další nevýhoda tohoto přístupu byla závislost podoby výchozích bodů na trénovacím datasetu - výsledný model pak vykazoval nižší schopnost generalizace.

Pro výpočet lokalizační chyby SSD používá funkci smooth L1 (stejně jako Faster R-CNN). Ta sice obecně vykazuje nižší přesnost než L2, ale na druhou stranu dává větší prostor pro

manipulaci s boxem díky menší náchylnosti na odlehlá pozorování - tedy zanedbatelné detaily v obrázku.

SSD přidává k MultiBoxu klasifikaci. Výstupem pro každý box je tedy vektor odpovídající distribuci pravděpodobností pro jednotlivé třídy.

3.5 Porovnání modelů pro detekci objektů

Následuje porovnání výkonu metod popsaných v předchozích sekcích na obecných datasetech pro detekci objektů. Vzhledem k množství proměnných při učení a evaluaci modelu je nutno brát následující výsledky jako orientační. Pro reálné použití je třeba vždy najít správnou rovnováhu mezi rychlostí a přesností v závislosti na konkrétním případu užití. Obecně platí, že pokud potřebujeme nejvyšší možnou přesnost, volíme model založený na Faster R-CNN. Naopak pokud je hlavním ukazatelem rychlost, vhodným kandidátem je model SSD.[25]

Tabulka 2: Porovnání metod na datasetu PASCAL VOC2007. Poslední dva sloupce tabulky značí počet vygenerovaných návrhů regionů a rozlišení vstupního obrázku.[6]

Model	mAP	FPS	Boxů	Vstup
Faster R-CNN	0,73	7	∼6000	∼1000x600
Fast YOLO	0,52	155	98	448x448
YOLO	0,66	21	98	448x448
SSD300	0.74	46	8732	300x300
SSD512	0.76	19	24564	512x512

Tabulka 3: Porovnání metod na datasetu MS COCO. Sloupec 0,5:0,95 značí zprůměrovanou metriku mAP pro hodnoty IoU v daném intervalu. Následující sloupce udávají mAP pro IoU 0,5 a 0,75.[7]

Model	0,5:0,95	0,5	0,75
Fast R-CNN	0,20	0,40	0,19
Faster R-CNN	0,24	0,45	0,23
YOLO	0,21	0,44	0,19
SSD300	0.23	0,41	0,23
SSD512	0.21	0,44	0,19

4 Detekce textu

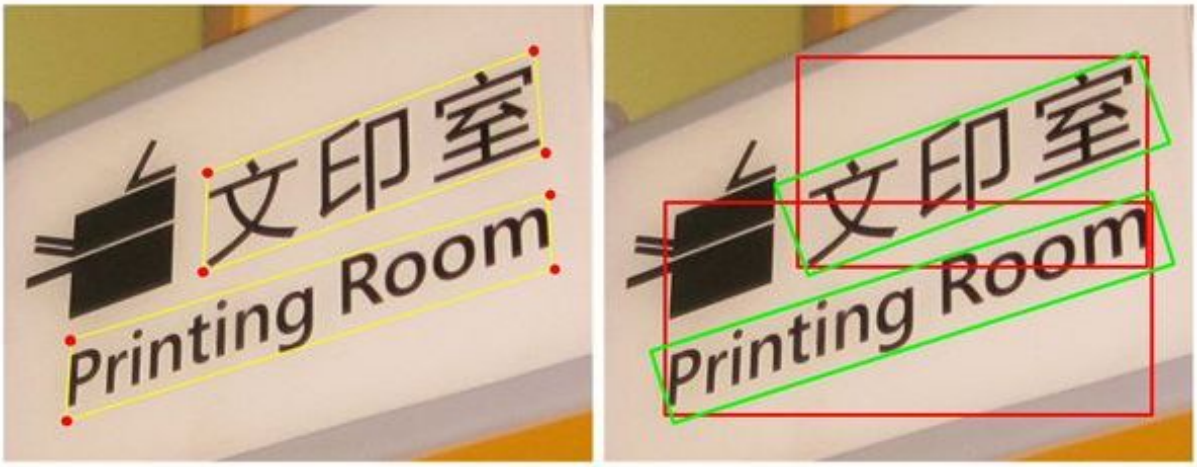
Všechny výše uvedené modely byly navrhnuty za účelem detekování objektů v obrázcích. Na detekci textu se můžeme dívat jako na podtřídu tohoto problému a tudíž pro jeho detekci zmíněné modely použít. Na druhou stranu má ale text oproti jiným objektům jistá specifika. Tyto vlastnosti se pak projeví ve fázi lokalizace, tedy ohraničení textu. Následující metody vycházejí ze základních modelů pro detekci objektů, ale obsahují úpravy, které jím umožní využít těchto specifických vlastností textu a tím zvýšení přesnosti.

4.1 Problémy specifické pro detekci textu

Poměr stran Na rozdíl od obvykle detekovaných objektů se text může vyskytovat v libovolném poměru stran. Například jeden dlouhý řádek by mohl mít rozměry 10x300px, jeden odstavec 300x300px a vertikálně psaný text pak 300x10px. Klasické modely pro detekci objektů pracují s obdélníkovými boxy pro označování objektů. Parametry těchto boxů (velikost a poměr stran) se obvykle nastavují ručně podle řešeného problému. Například Faster R-CNN používá ve výchozí konfiguraci 9 boxů – kombinace 3 rozměrů a 3 poměrů stran. Tento přístup funguje velmi dobře pro obecné objekty, ale vzhledem k množství tvarů a velikostí, v nichž se může vyskytovat text, není dostatečně flexibilní pro jeho detekci.

Tvar Další problém pak nastává v deformaci textu způsobené úhlem pohledu. Například při fotografování nápisu na stěně výsledný tvar textu na fotografii velmi závisí na pozici fotoaparátu. Příklad problémového označení textu je znázorněn na obrázku (7).Všechny výše popsané metody jsou založeny na označování detekovaných objektů obdélníkem orientovaném rovnoměrně s hranami obrázku. Se zvyšujícím se úhlem pořízení fotografie text ztrácí obdélníkový tvar, a pro jeho přesné označení bychom pak potřebovali jiný čtyřúhelník. Přesné označení textu otočeného např. o 45° by rovněž znamenalo problém.

V podstatě tyto modely dosahují vysoké přesnosti pouze pro horizontální texty, ideálně o malém počtu znaků - např. nápisy na značkách, budovách apod. Jakmile se objeví text nakloněný, zkosený vlivem perspektivy, nebo třeba jen velmi dlouhý, IoU označení rovným obdélníkem, a s ním logicky i přesnost celého modelu, značně klesá. K vypořádání se s tímto problémem existuje celá řada metod, jak upravit systém navrhování boxů tak, aby byl schopen navrhovat obdélníky natočené pod potřebným úhlem, případně místo obdélníků použít jiný čtyřúhelník.



Obrázek 7: Příklad problémové detekce textu. Vlevo je zobrazeno ideální ohraničení textu. Vpravo pak označení pomocí obdélníků rovnoběžných k okrajům obrázku (červeně) a pomocí orientovaných obdélníku (zeleně).

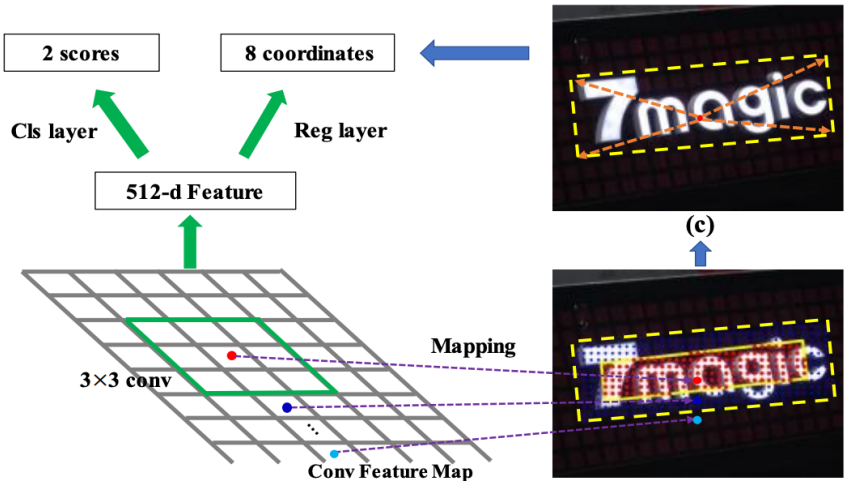
4.2 AF-RPN

Jedná se o přizpůsobenou verzi originální RPN pro Faster-RCNN model. Namísto původního systému posuvného okénka, kde se pro každou pozici okénka generovalo n návrhů boxů podle přednastavených kriterií (různé velikosti a poměry stran), se zde používá přístup posuvného bodu. To znamená že od každého bodu, který se nachází v ground-truth boxu, se predikuje vzdálenost k okrajům příslušného boxu. Tímto způsobem je možné generovat boxy různých tvarů a velikostí, bez nutnosti přednastavovat tyto parametry ručně.

Princip

Základem AF-RPN[26] je síť FPN[27] postavená na ResNet50[24] a tři moduly pro detekci. Síť ResNet50 je odpovědná za vytvoření feature map. Detektory jsou pak připojeny na tři feature mapy různých rozměrů. Detekce probíhá tak, že se každý bod feature mapy namapuje na posuvný bod v původním obrázku. Postup detekce je znázorněn na obrázku (8). Pro tento posuvný bod se pak predikuje třída (obsahuje text/neobsahuje text) a jeho posun oproti každému ze čtyř bodů, jimiž je definován ground-truth box.

V obdélníkových ground-truth boxes datasetů pro detekci textu se obvykle nachází šum (pozadí) vlivem nedostatečně přesného ohraničení textu. Proto při učení vytvoříme *jádro* pro každý ground-truth box tak, že vynásobíme jeho kratší stranu konstantou 0,5 a delší stranu konstantou 0,8. Jádro tak sice pravděpodobně nebude obsahovat celý text, ale naopak většina prostoru jádra by měla být vyplněna textem. Při učení poté posuvné body uvnitř jádra považujeme za pozitivní vzorky. Posuvné body uvnitř ground-truth boxu, ale vně jádra, ignorujeme a body vně ground-truth boxu považujeme za negativní vzorky.



Obrázek 8: Schéma AF-RPN. Každý bod feature mapy se namapuje na posuvný bod v původním obrázku. Na pravém spodním obrázku je zobrazeno jádro boxu (body zbarvené červeně). Vpravo nahoře jsou zobrazeny vektory posunu bodu vzhledem k rohům ground-truth boxu (žlutý přerušovaný obrys). [26]

Feature Pyramid Network (FPN)

Konvoluční sítě fungují jako feature extractor. To znamená že (pří průchodu *sdola nahoru*) redukují dimenzionalitu (v kontextu zpracování obrazu rozlišení) vstupu tak, aby zachovaly nejvýznamnější informace. Čím nižší rozlišení feature mapa má, tím větší má však každý její bod receptivní pole (odpovídá většímu počtu pixelů vstupního obrázku, tedy obsahuje větší množství informací). Myšlenka FPN je tedy rozšířit koncept Residual Networks[24] a přidat průchod *shora dolů*. V jeho průběhu bude cílem vytvořit feature mapy různých rozlišení, které však budou obsahovat informace ze všech konvolučních vrstev, získané při průchodu *zdola nahoru*. Průchod *shora dolů* probíhá následovně:

1. Feature map s nižším rozlišením se zvětší metodou nejbližšího souseda [28]
2. Na feature map z průchodu *zdola nahoru* s odpovídajícím rozlišením se aplikuje konvoluce 1x1, která zajistí redukci její hloubky
3. Tyto dvě mapy se sečtou

Výsledná síť FPN minimalizuje hlavní nedostatek Residual Network, a sice detekci malých objektů.

4.3 TextBoxes

Jako metoda odvozená z modelu SSD se TextBoxes [29] vyznačuje nízkou výpočetní náročností a tím pádem vysokou rychlostí. Rozdíl oproti SSD spočívá v přidání text-box vrstev. Autoři zde počítají s tvarem textu, kde se šířka rovná několikanásobku výšky, a podle toho nastavují

poměry stran (hodnoty 1, 2, 3, 5, 7, 10). Dále se v text-box vrstvách používají konvoluční filtry o netradičním rozměru 1x5, místo standardních 3x3. Toto nastavení zaručuje, že šířka generovaných feature map budou, stejně jako u textu, několikanásobkem jejich výšky.

Architektura

Základ architektury je opět tvořen sítí VGG-16[19], přičemž její plně propojené vrstvy jsou odstraněny. Na jejich místě je přidáno 9 konvolučních vrstev postupně zmenšujících se rozměrů. Na konci modelu se nachází 6 text-box vrstev, kde každá je napojena na jednu z předchozích 9 konvolučních vrstev.

Celý model je tedy reprezentován jednou 28 vrstvou plně konvoluční sítí, z čehož vyplývá, že přijímá na vstupu obrázky libovolných rozměrů. Na výstup modelu se tradičně aplikuje metoda non-max suppression (NMS).

Text-box vrstvy

Text-box vrstvy zajišťují současnou predikci boxů a prezenci textu. Stejně jako v SSD tyto vrstvy operují na feature mapách různých rozměrů a jejich vstup je definován výchozími boxy. Rozměry výchozích boxů počítají s horizontálními texty. Jejich šířka by tedy měla být několikanásobkem jejich výšky. V publikaci autoři volá poměry stran 1, 2, 3, 5, 7 a 10. Díky tomu jsou výchozí boxy na mřížce uspořádány velmi hustě v horizontálním směru, ale řídce ve vertikálním. Abychom tento nedostatek odstranili, používáme pro boxy vertikální posun. Každý rozměr výchozího boxu je tedy obsažen v každé buňce mřížky dvakrát. Z toho jednou je jeho posun nulový, a jednou odpovídá polovině výšky buňky.

Velikost kernelů v text-box vrstvách volíme 1x5, čímž opět zdůrazňujeme, že hledáme velmi široké objekty. Tyto kernely pak budou mít velmi široké receptivní pole, tudíž lépe pokryjí dlouhé texty a zároveň se vyhnou šumu nad a pod textem.

4.4 Výsledky metod při detekci textu

Následující porovnání v tabulce [4] bylo vyhodnocováno v rámci ICDAR 2013 Robust Reading Competition [30]. Konkrétně se zde jednalo o první úkol - lokalizace textu. Dataset pro tento úkol obsahoval 229 fotografií a celkem 848 slov.

Tabulka 4: Porovnání metod na datasetu ICDAR 2013. AF-RPN zde znamená Faster R-CNN, kde je původní RPN nahrazena AF-RPN.[22][26]

Model	Recall	Precision	F-measure
SSD	0.80	0.60	0.68
Faster R-CNN	0.75	0.71	0.73
TextBoxes	0.88	0.83	0.85
AF-RPN	0,90	0,94	0,92

5 Experimenty

Pro experimenty jsem vybral jednoho zástupce pro každou z hlavních kategorií detektorů. Ze skupiny single-stage metod jsem zvolil Faster R-CNN, jelikož se jedná o stage-of-the-art model ve své kategorii z hlediska přesnosti. Na druhou stranu jako zástupce two-stage metod používám model SSD, který dominuje v detekci objektů především z hlediska rychlosti.

Veškeré výpočty provádím za pomocí frameworku Tensorflow[31] a jeho nadstavbou Object Detection API[32]. Tato nadstavba poskytuje implementaci předních modelů pro detekci objektů, rozhraní pro jejich trénování a v neposlední řadě také metriky pro jejich evaluaci. Pro psaní veškerých skriptů potřebných k experimentům jsem použil programovací jazyk Python, při práci s datasety pak využívám nástroje Jupyter Notebook[33]. Konfigurace definující model a proces jeho učení definuji pomocí protokolu Protocol Buffers[34]. Jako prostředek pro vizualizaci používám Tensorboard[35].

Ověření správnosti formátu datasetu a konfigurace modelu provádím lokálně, případně v cloudu pomocí nástroje Google Collaboratory[36]. Po ověření správnosti spouštím trénovací proces pomocí služby AI Platform, což je součást platformy Google Cloud Platform[37], zaměřující se na výpočty pro strojové učení.

5.1 Dataset

Základem každé neuronové sítě je kvalitní dataset. V oblasti detekce objektů je však výroba datasetu velmi náročná a tak zde panuje obecný nedostatek dat. Pro mnou řešený problém potřebuji dataset obsahující ideálně fotografie z reálného prostředí, na kterých budou zachyceny nápisy v Japonštině, Čínštině, čí Korejštině. Jediný dataset splňující tyto požadavky je vyvíjen v rámci mezinárodní konference pro analýzu a rozpoznávání dokumentů ICDAR. Každé dva roky se zde vyhlašují výzvy na různá témata spojená se zpracováním textu a současně k těmto výzvám bývá vydána řada datasetů. Pro účely této práce používám dataset vydaný k výzvě robustního čtení v roce 2017 a 2019 [30].

Vzhledem k počtu možných variací hledaného textu je však pravděpodobné, že tento dataset nebude pro naučení dostačovat. Jazyky, mezi kterými se model bude snažit rozpoznávat, obsahují dohromady několik desítek tisíc znaků. Rozdíly mezi jednotlivými znaky jsou navíc často zanedbatelné. Je tedy jasné, že čím více dat síti poskytneme, tím větší máme šanci obdržet přijatelný výsledek. Proto se v této práci pokusím vygenerovat syntetický dataset, který by teoreticky měl pomoct síti rozpoznávat jednotlivé tvary znaků. Postup předzpracování ICDAR datasetu a generování umělého datasetu je popsáno v následujících sekcích.

5.1.1 Dataset ICDAR

Pro experimenty jsem použil dataset k úkol Mulit-lingual Text Detection (MLT) ve verzi ICDAR 2017 a ICDAR 2019. Ten obsahuje reálné fotografie s nápisy v různých jazycích. Ke každému

Obrázek 9: Ukázka datasetu odvozeného z ICDAR. Červené ohraničení znázorňuje původní čtyřúhelníkové boxy. Žlutě jsou vyznačeny z nich odvozené obdélníkové boxy, které používám pro učení modelů. Na obrázku vpravo lze vidět obtížnost tohoto datasetu způsobenou velmi malými rozměry objektů zájmu. Na některých obrázcích se vyskytuje až několik desítek miniaturních slov. V této fázi zůstávají označeny i třídy, o které nemám zájem - např. text v latince a čísla v levém obrázku. Tyto boxy odstraňuji až při převodu datasetu do formátu TF record. Převod je podrobněji popsán v následujících sekcích.

nápisu je pak přiřazen ohraničující box, třída odpovídající názvu písma a přepis textu. Celkem je zde obsaženo 10 tříd, včetně Čínštiny, Korejštiny a Japonštiny, které nás budou jako jediné zajímat. Dataset z roku 2017 zahrnuje 7200 obrázků pro trénování a 1800 pro evaluaci. Novější verze pak obsahuje 10000 obrázků pro trénování. Evaluační dataset pro tuto verzi nebyl dosud vydán.

Úpravy datasetu

Původní rozdělení datasetu vzhledem k povaze úkolu nerespektuji. Všechny tři části spojím do jednoho datasetu, který po patřičných úpravách rozdělím na trénovací a evaluační podle potřeby. Jelikož tyto datasety obsahují více informací, než pro tuto práci potřebuji, nejprve z nich vyberu jen nezbytná data. Odstraním tedy všechny obrázky, které neobsahují žádnou ze tří tříd zájmu. Po této filtraci však dataset stále obsahuje instance textu obsahující znaky latinky, či číslice. Takové instance jsou pro můj model nežádoucí, protože by jej mohly ovlivnit k označování nechtěného textu. Odstranění takto závadných obrázků však výrazně zasáhne do celkové velikosti datasetu. Proto raději vytvořím dvě verze - jednu s odstraněním obrázků obsahujících takto problémová slova a druhou bez jejich odstranění.

Výsledný dataset rozdělím na trénovací a evaluační v poměru 5:1. Pro evaluaci je klíčová vyváženost tříd, která danému datasetu chybí. Při rozdělování tedy postupuji tak, že náhodně vybírám soubory z každé třídy tak dlouho, dokud počet instancí dané třídy neodpovídá jedné pětině počtu instancí nejméně početné třídy. Takto vytvořený evaluační dataset je vyvážený, ale

zbytek datasetu nikoliv. Dále tedy postupuji obdobným způsobem. Tentokrát náhodně odstraňuji obrázky obsahující alespoň jednu ze dvou nejvíce zastoupených tříd tak dlouho, dokud počet instancí obou těchto tříd neodpovídá počtu instancí nejméně zastoupené třídy. Vyvažování bez vytváření umělých vzorků však znamená další snížení velikosti datasetu. Pro trénovací dataset tedy opět vytvořím dvě verze - jednu vyváženou, druhou původní.

Poslední změnou, kterou musím učinit je převod tvaru boxu ohraničujícího text. V použitém datasetu je box zadán čtyřmi body představujícími rohy čtyřúhelníku libovolného tvaru. Modely, které budu používat pro učení však jsou schopny pracovat pouze s boxy ve tvaru obdélníku rovnoběžného s hranami obrázku. Proto je nutné změnit formát tohoto ohraničení. Při převodu postupuji tak, že vyhledám pro x a y jejich nejmenší a největší hodnoty. Dvojice (x_{min}, y_{min}) a (x_{max}, y_{max}) pak odpovídají protilehlým rohům obdélníku. Ukázka z datasetu ICDAR je zobrazena na obrázku (9). Tabulka (5) zobrazuje výsledné počty souborů a instancí pro jednotlivé třídy a verze datasetu.

Tabulka 5: Datasety odvozené z ICDAR. Trénovací opt. znamená dataset s odstraněnými problémovými slovy a s vyvážením tříd. Instance odpovída v datasetu jednomu slovu.

	Čínština		Japonština		Korejština	
Dataset	Souborů	Instancí	Souborů	Instancí	Souborů	Instancí
Trenovací	1699	5752	1866	12310	1944	14019
Trénovací opt.	1583	5022	1081	5142	926	5244
Evaluační	286	893	200	935	177	1009

5.1.2 Syntetický dataset

Jak už jsem zmínil v předchozí sekci, dataset ICDAR pravděpodobně nebude svým rozsahem dostačující. Navíc jeho verze ICDAR 2019 byla vydána až v březnu 2019. Na začátku práce jsem tedy měl k dispozici pouze verzi ICDAR 2017. Z toho důvodu jsem se rozhodl vytvořit skript na generování umělého datasetu prakticky libovolné velikosti.

Jako pozadí používám obrázky z testovacího datasetu k výzvě Open Images 2018 [38]. Jedná se o dataset pro detekci objektů o velikosti 100 000 obrázků. Na toto pozadí následně generuji náhodné skupiny znaků žádaných jazyků. Celkem používám pro text 4 typy písma:

- Hangul (Korejština)
- Hiragana (Japonština)
- Katakana (Japonština)
- Čínština

Tabulka 6: Písma použitá pro generování datasetu.

Písmo	UNICODE rozsah	Počet znaků	Třída
Hangul	0xAC00 - 0xD7A3	11 171	Korean
Hiragana	0x3041 - 0x309F	94	Japanese
Katakana opt.	0x30A0 - 0x30FF	95	Japanese
Čínština	0x4E00 - 0x9FCF	20 943	Chinese

Přestože se čínské písmo využívá i k psaní v korejském, či japonském jazyce, považuji ho vždy za čínštinu. V této práci se zabývám výhradně vizuálním rozpoznáváním písma, není tedy možné rozlišovat jazyk podle kontextu.

Detaily použitých písem a jejich reprezentací v tabulce UNICODE jsou zobrazeny v tabulce (6). Pro vykreslení textu do obrázku používám font *SourceHanSans-Regular.ttc*, který vydává pod licencí *SIL Open Font License* společnost Adobe.

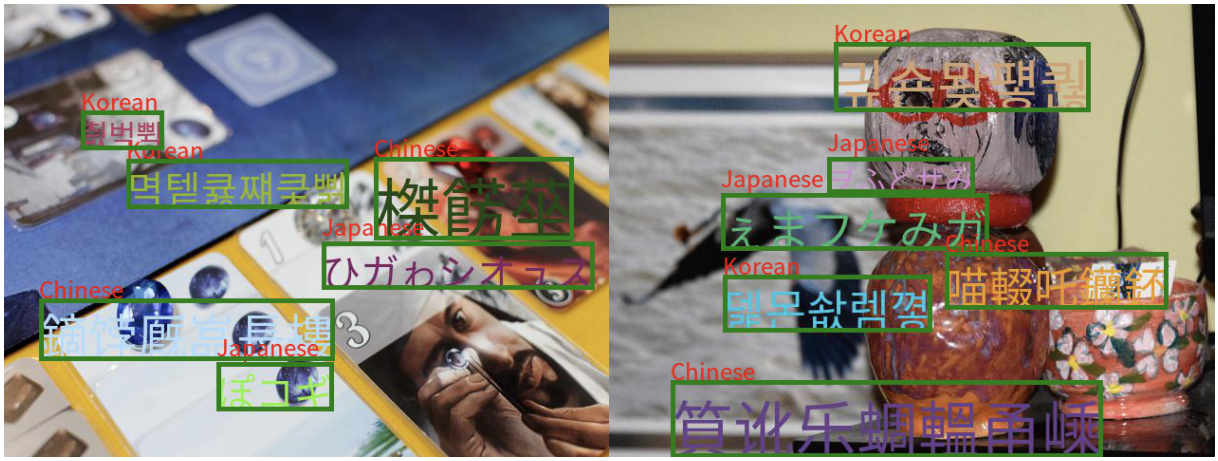
Postup generování

Metoda pro generování datasetu přijímá 4 argumenty:

- Pole polí obsahujících znaky jednoho jazyka
- Počet obrázků
- Rozměry výsledných obrázků
- Počet slov (náhodných uskupení symbolů abecedy) pro každý jazyk a obrázek
- Minimální a maximální délku slova
- Minimální a maximální velikost písma

V prvním kroku generátor náhodně vybere podmnožinu obrázků o velikosti zadané parametrem. Jako zdroj pro tyto obrázky slouží výše zmíněný dataset Open Images[38]. Následuje iterování přes všechny takto vybraná pozadí. Pokud je obrázek otočen na výšku, přetočí se o devadesát stupňů, abychom zajistili, že jeho šířka je větší než výška. Všechny obrázky se transformují na fixní velikost zadanou parametrem. Díky tomu je možné zpracovávat při učení více obrázku najednou pomocí tzv. mini-batch techniky. Otočení obrázku tedy zmírňuje deformaci při změně jeho velikosti.

Následně probíhá vykreslení samotného textu do obrázku. Nejprve se náhodně vygeneruje slovo, tedy náhodný sled symbolů jedné ze zadaných sad. Délka slova a velikost fontu je rovněž náhodně vygenerovaná v zadaném rozsahu. Konkrétně používám hodnoty 3-7 pro délku slova a 30-70 pro velikost fontu. Poté se slovo umístí na náhodnou pozici v původním obrázku. Po



Obrázek 10: Ukázka syntetického datasetu. Tento dataset neobsahuje tak komplikované instance textu, jako např. velmi malé, či zdeformované texty v datasetu ICDAR. Na druhou stranu se zde často vyskytuje text v minimálním kontrastu s pozadím. Například světle modrý čínský text na levém obrázku je pro člověka obtížně čitelný.

umístění textu se zkontroluje, zda se jeho pozice nepřekrývá s jiným textem vygenerovaným v předchozích iteracích. Pokud ano, text se zahodí a proces se opakuje. V opačném případě se uloží záznam o vykreslení textu do objektu, jenž nakonec slouží pro vytvoření souborů s anotacemi, a pokračuje se dalším slovem. Obrázek (10) zobrazuje ukázkový vzorek z výsledného datasetu. Po vykreslení všech zadaných slov na obrázek se spouští generování anotací. Ty se ukládají do XML souboru ve formátu PASCAL VOC[14]. V tomto souboru se uchovávají informace jak o obrázku (velikost, počet kanálů, ...), tak o všech objektech, které obsahuje (pozice, rozměry, třída). Po dokončení všech obrázků se vygeneruje soubor *label_map.pbtxt*, jenž slouží pro mapování názvu tříd na odpovídající numerický identifikátor.

5.2 Učení

Cílem této práce je najít co možná nejlepší způsob detekce a rozpoznání asijských textů. Vzhledem k tomu, že dosud není znám způsob, který by vedl k zaručeně nejlepšímu modelu, zkouším v mých experimentech několik variant, jejichž výsledky mohu následně porovnat. Na různých modelech tak mohu zkoumat vliv použitého datasetu, odlišných metod předzpracování, nebo třeba různé konfigurace hyperparametů modelu.

Pro učení modelů používám různé kombinace datasetu ICDAR s vlastním vygenerovaným. Celkem jsem vytvořil čtyři datasety pro trénování:

- **Syntetický** - obsahuje 200 000 instancí z každého jazyka
- **ICDAR** - obsahuje cca 12 000 japonských slov, 5 700 čínských a 14 000 korejských

- **ICDAR optimalizovaný** - odstraněny obrázy obsahující slova s latinkou/číslicemi, vyvážená čestnost instancí všech tříd. Po optimalizaci rozdělen na trénovací a evaluační část v poměru 5:1. Obsahuje cca 5000 instancí každého jazyka.
- **Kombinovaný** - spojení ICDAR optimalizovaného a generovaného (pro zachování vyvážení pouze 2500 vygenerovaných obrázků - 5000 instancí každého jazyka). Obsahuje cca 10 000 instancí každé třídy

Pro evaluaci rovněž používám různé varianty:

- **Syntetický** - obsahuje 900 instancí z každého jazyka
- **ICDAR optimalizovaný** - evaluační část odebrána z optimalizovaného ICDAR datasetu. Obsahuje cca 900 instancí z každého jazyka.
- **Kombinovaný** - spojení předchozích dvou datasetů

Každý model vždy trénuji na jednom z trénovacích datasetů. Používám mini-batch o velikosti 8. V průběhu učení se každých deset minut spouští evaluace na všech třech evaluačních datasetech. Výsledky této evaluace se ukládají a po zobrazení v nástroji Tensorboard[35] slouží k analýze modelu. Vždy trénuji dva modely se stejným datasetem a natavením - Faster R-CNN a SSD. Díky toho můžu porovnat chování single-stage a two-stage modelů. Na doporučení autorů těchto metod vždy používám feature extractor předučený na obecném datasetu. Tyto předučené modely jsou k dispozici v rámcí Object Detection API. Konkrétně jsem vybral *faster_rcnn_resnet101_coco* pro Faster R-CNN a *ssd_resnet_50_fpn_coco* pro SSD. Jak už název napovídá, tyto modely jsou natrénovány na COCO datasetu.

Pro optimalizaci algoritmu gradient descent používám momentum. Koeficient momentum nastavuji na hodnotu 0.9. Dále u modelů používám strategii pro optimalizaci koeficientu učení *cosine decay* s mezními hodnotami 0.13 a 0.04. Koeficient učení tedy zpočátku velmi rychle vystoupá z původní hodnoty 0.13 na 0.04 a poté pozvolna klesá. Pokud se na celkové chybě modelu při blížení koeficientu učení k nule projeví, že model přestává konvergovat právě vlivem téměř nulového koeficientu, nastavím nový koeficient učení konstantou. Tuto hodnotu vybírám z předchozích hodnot podle toho, při které model nejrychleji konvergoval. Následně už koeficient měním manuálně vždy, když se začnou objevovat v celkové chybě velké výkyvy.

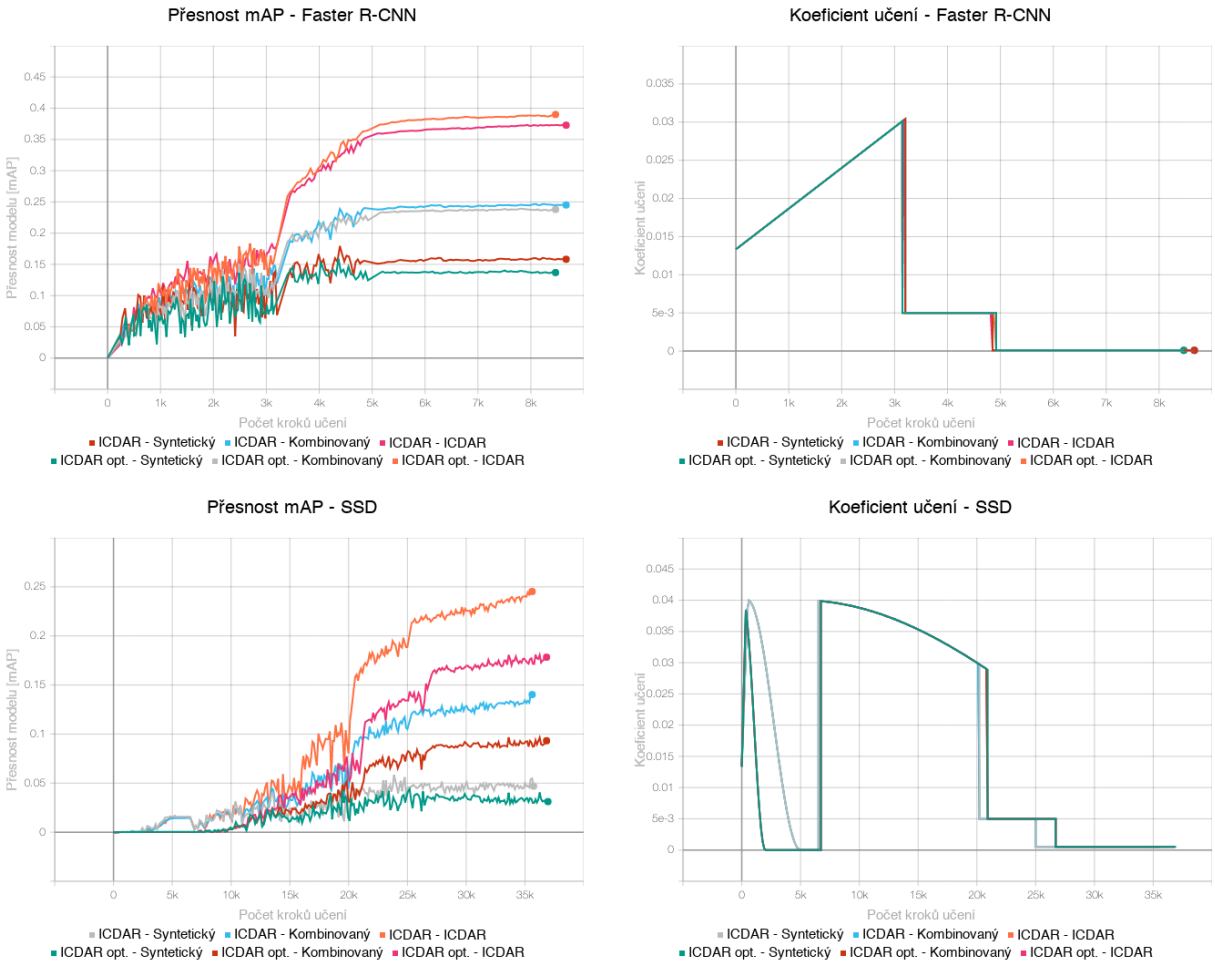
Učení na syntetickém datasetu

Jako první jsem začal trénovat na vygenerovaném datasetu. Zde jsem měl jistotu, že dat je dostatek. První dva modely tak měly ukázat, zda je vůbec možné úspěšně síť natrénovat na tak rozmanitou množinu objektů (desetitisíce znaků pro jednu třídu). Průběh vývoje přesnosti mAP a celkové chyby modelu je zobrazen na obrázku (11). Z těchto grafů lze vyčíst následující postřehy:

Obrázek 11: Průběh učení na syntetickém datasetu. Jeden krok odpovídá průchodu osmi obrázků a následné upravě vah.

- Přesnost na syntetickém datasetu se zvyšuje velmi prudce až ke hranici 70% mAP. To je v oblasti detekce objektů velmi vysoká hodnota. Můžeme tedy říci, že samotné naučení rozpoznávání jazyka podle vizuální podoby jeho znaků není pro tento model problém.
- Naopak chyba na datasetu ICDAR prakticky ani nezačala konvergovat.
- Chyba na kombinovaném datasetu se pak logicky nachází zhruba uprostřed mezi dvěma výše zmíněnými. Pravděpodobně se tedy správně vyhodnocuje pouze jeho syntetická část.

Syntetický dataset se zdá být pro učení modelů velmi jednoduchý, vzhledem k tomu že dosáhly velmi vysoké přesnosti během tisíce kroků (průchodu osmi tisíc obrázků), což je jen zlomek celkové velikosti datasetu. Po dosažení 70-80% mAP začala přesnost obou modelů výrazně kolísat. Dá se tedy předpokládat, že snížením koeficientu učení by se dala výsledná přesnost ještě zvýšit. V experimentech s těmito modely jsem již nepokračoval, jelikož z hlediska použitelnosti bylo vhodnější se zaměřit na přesnost na reálném datasetu ICDAR.



Obrázek 12: Průběh učení na reálném datasetu. Legenda je značena ve tvaru trénovací set - evaluační set. ICDAR opt. označuje optimalizovanou verzi datasetu (promazání a vyvážení)

Tento experiment ukázal, že modely pro detekci objektů jsou schopny naučit se detekovat text a určit jeho jazyk podle vizuální podoby jeho symbolů a to v relativně krátkém čase, navíc za použití malého vzorku dat v poměru k celkovému počtu různých znaků. Zároveň se však nepodařilo prokázat možnost reálná data zcela nahradit daty syntetickými.

Učení na reálném datasetu

Jak jsem zmiňoval v sekci popisující přípravu datasetů, ICDAR dataset z reálného prostředí používám ve dvou verzích. První z nich je zaměřená na kvantitu - používám všechny obrázky obsahující instance žádaných tříd. Druhá verze spoléhá na kvalitu dat - je jich sice výrazně méně, ale neobsahují problémová slova a třídy mají vyváženou četnost výskytu. Průběh učení je zachycen na obrázku (12. Při učení těchto modelů jsem experimentoval s koeficientem učení. Především na modelu SSD se výrazně projevila špatná volba počátečních parametrů strategie

cosine decay. Vlivem příliš krátkého intervalu změny (2000 kroků) se koeficient příliš rychle přiblížil nule a učení modelu tak stagnovalo. Po zvýšení intervalu na 50 000 kroků se chování modelu značně zlepšilo (změnu jsem provedl okolo hranice 7 000 kroků). Zhruba po 5 000 krocích od této změny se začala projevovat příliš velká hodnota koeficientu a křivka chyby se rozkmitala. Zhruba v kroku 20 000 jsem pak přešel na konstantní hodnotu koeficientu, což odstartovalo prudké zvyšování přesnosti. Poslední změnu jsem provedl na kroku 27 000, kde jsem snížil hodnotu na deset procent původní hodnoty, tedy 0,0001. Takto jsem pokračoval v učení, dokud model nepřestal konvergovat.

V případě Faster R-CNN jsem už postupoval přímočařeji. Nastavil jsem od začátku dlouhý interval pro změnu (50 000 kroků). Tím jsem zajistil pomalé zvyšování koeficientu. Po třech tisících krocích jsem z vizualizace vybral hodnotu koeficientu, kdy funkce mAP stoupala nejstabilněji. Poté co se funkce opět rozkmitala jsem znovu snížil hodnotu o devadesát procent.

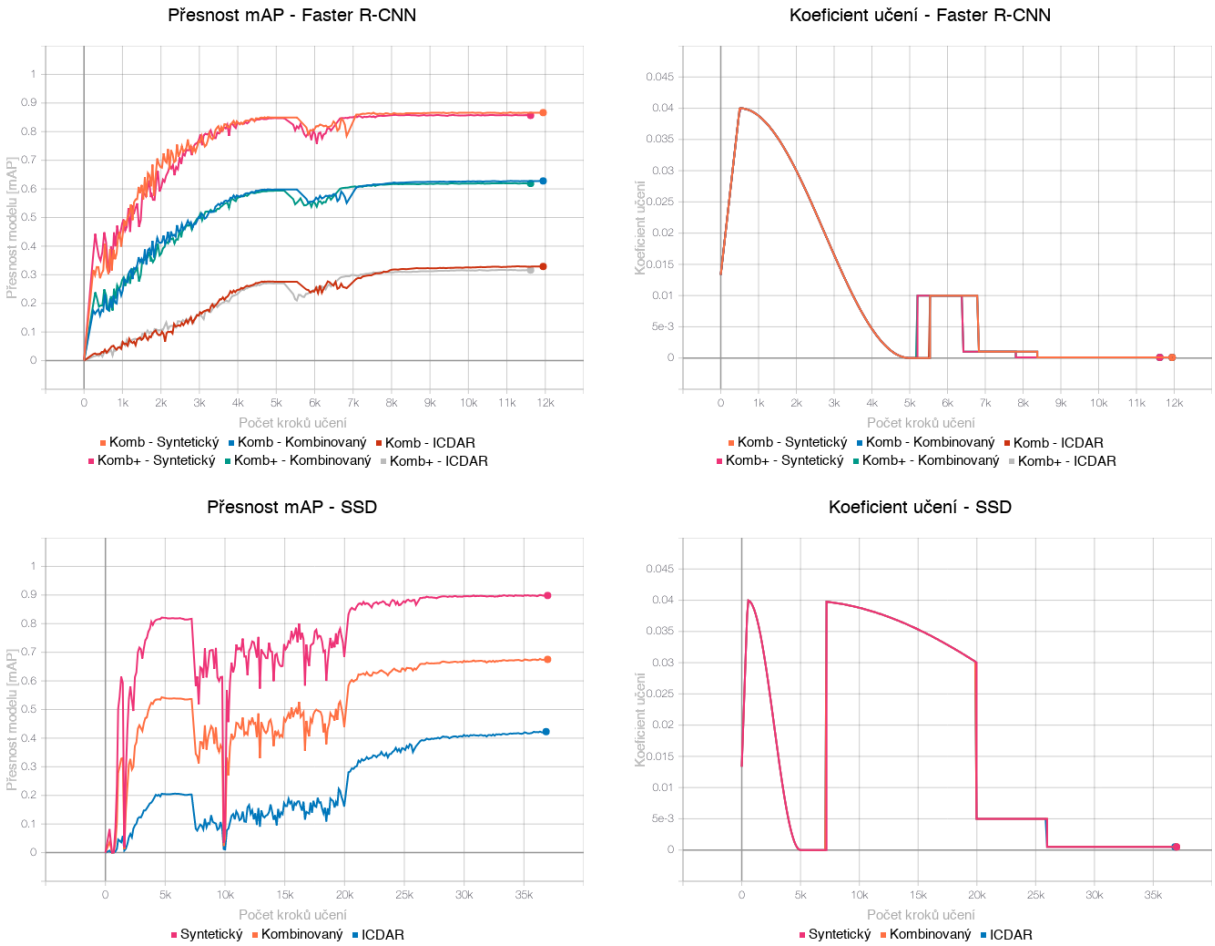
Konvergence na těchto datasetech trvala několikanásobně déle v porovnání s těmi syntetickými, a dosažené výsledky zdaleka nedosahují takové kvality. Při tomto experimentu se již naplno projevily rozdíly mezi single-stage a two-stage modely. Faster R-CNN dosáhl výrazně vyšší přesnosti (39 % mAP) a ke konvergenci mu stačilo zhruba 8 500 kroků. U modelu SSD můžeme odečíst prvních dest tisíc kroků, v jejichž průběhu model nekonvergoval vlivem špatného nastavení. I přesto však model Faster R-CNN dosáhl značně lepšího výsledku za třetinu průchodů datasetu. Na druhou stranu model SSD dominuje svou rychlostí průchodů. Průchod 35 000 kroků mu trval 53 hodin, zatímco Faster R-CNN prošel 8 500 kroků za 36 hodin. To v přepočtu znamená zpracování 0,5 obrázku za sekundu pro Faster R-CNN, a 1,5 obrázku za sekundu pro model SSD.

Samotná úprava datasetu se při evaluaci na reálném setu projevila zvýšenou přesností optimalizované verze o jeden a půl procentního bodu. U modelu SSD je rozdíl nižší, zde je však třeba brát v úvahu celkově nižší hodnoty přesnosti. Za zmínku také stojí, že ke zvýšení přesnosti došlo pouze při evaluaci na reálném datasetu, u zbylých dvou došlo naopak k mírnému poklesu. Při těchto evaluacích model pravděpodobní trpí nedostatkem generalizace spojeným s výrazným zmenšením datasetu vlivem optimalizace.

Celkový výsledek (především modelu Faster R-CNN) je příjatelný, vzhledem k tomu, že modely pro detekci objektů na obecných datasetech jen výjimečně překročí 50 % mAP[39][40][41]. Zajímavý je také fakt, že modely konvergovaly i z hlediska přesnosti na syntetickém datasetu. Zatímco při snaze naučit model reálný dataset pomocí syntetického jsem se nedostal ani na jedno procento mAP, při opačném postupu, tedy učení na syntetická data pomocí dat reálných, jsem se dostal na bezmála šestnáct procent.

Učení na kombinovaném datasetu

Při posledních experimentech se věnuji učení modelů na datasetu složeném jak z reálných obrázků, tak ze syntetických. Průběh učení je zobrazen na obrázku (13). Na tomto datasetu jsem



Obrázek 13: Průběh učení na kombinovaném datasetu. Model Faster R-CNN označen jako Komb+ používá při trénování umělé rozšiřování dat pomocí několika metod předzpracování.

trénoval jeden Faster R-CNN model navíc. Na tom jsem testoval možnosti umělého rozšíření datasetu pomocí několika metod předzpracování. Konkrétně jsem konkrétně následující úpravy:

- Náhodný převod obrázku do odstínů šedé s pravděpodobností 0,1.
- Náhodná změna jasu o -20 % až 20 %
- Náhodná změna kontrastu o -20 % až 25 %
- Náhodná změna odstínu o -20 % až 20 %
- Náhodná změna nasycení o -20 % až 25 %
- Náhodná distorze barev

Samotné úpravy v rámci předzpracování měly na učení modelu prakticky nulový efekt. To nasvědčuje tomu, že modely nemají problém s detekcí v důsledku velké variace barev. Největší potíž pravděpodobně působí příliš mnoho možností tvarů textu. Nabízí se tedy otázka, zda by se dala zvýšit přesnost modelů použitím jiných metod předzpracování, které obrázky například různě deformují.

Celkové výsledky modelů trénovaných na kombinovaném datasetu však dopadly velmi dobře. Tyto modely vykazují největší průměrnou přesnost přes všechny tři evaluační datasety, což značí velkou schopnost generalizace. Verze Faster R-CNN sice ve výsledcích na reálném datasetu pokulhává za modelem učeným pouze na taková data o 6 procentních bodů, za to ale výrazně dominuje na zbývajících evaluačních datasetech. Verze SSD zde dokonce překonává všechny ostatní testované modely a s přesností 42 % mAP se ukazuje jako nejlepší volba pro úkol detekce a rozpoznání písma. Obzvláště pak vezmeme-li v potaz schopnost SSD zpracovávat desítky obrázku za sekundu, díky čemuž je tento model více než vhodný pro použití v reálném čase například v mobilních fotoaparátech.

5.3 Shrnutí dosažených výsledků

Následuje shrnutí výsledků jednotlivých modelů. V tabulce (7) je obsaženo pět modelů dosahujících nejvyšší přesnost pro každý použitý evaluační dataset. V poslední tabulce se pak nachází pět modelů s nejvyšší průměrnou přesností přes všechny evaluační datasety. Jednoznačně nejlepších výsledků dosáhl model SSD natrénovaný na datasetu složeném jak z reálných příkladů (podmnožina datasetu ICDAR), tak ze syntetických dat. Díky tomuto spojení model získal větší schopnost generalizace a ta se projevila na výsledcích všech evaluačních datasetů. Tento výsledek je poměrně překvapivý, vzhledem k tomu že modely SSD obvykle z hlediska přesnosti za Fater R-CNN pokulhávají, což si kompenzují několikanásobně větší rychlostí. Ovšem není to první případ, kdy model SSD dosáhl vyšší přesnosti. Už autoři jeho původní publikace zmiňují úspěch v podobě překonání přesnosti Faster R-CNN na datasetech COCO a PASCAL VOC 2007[6].

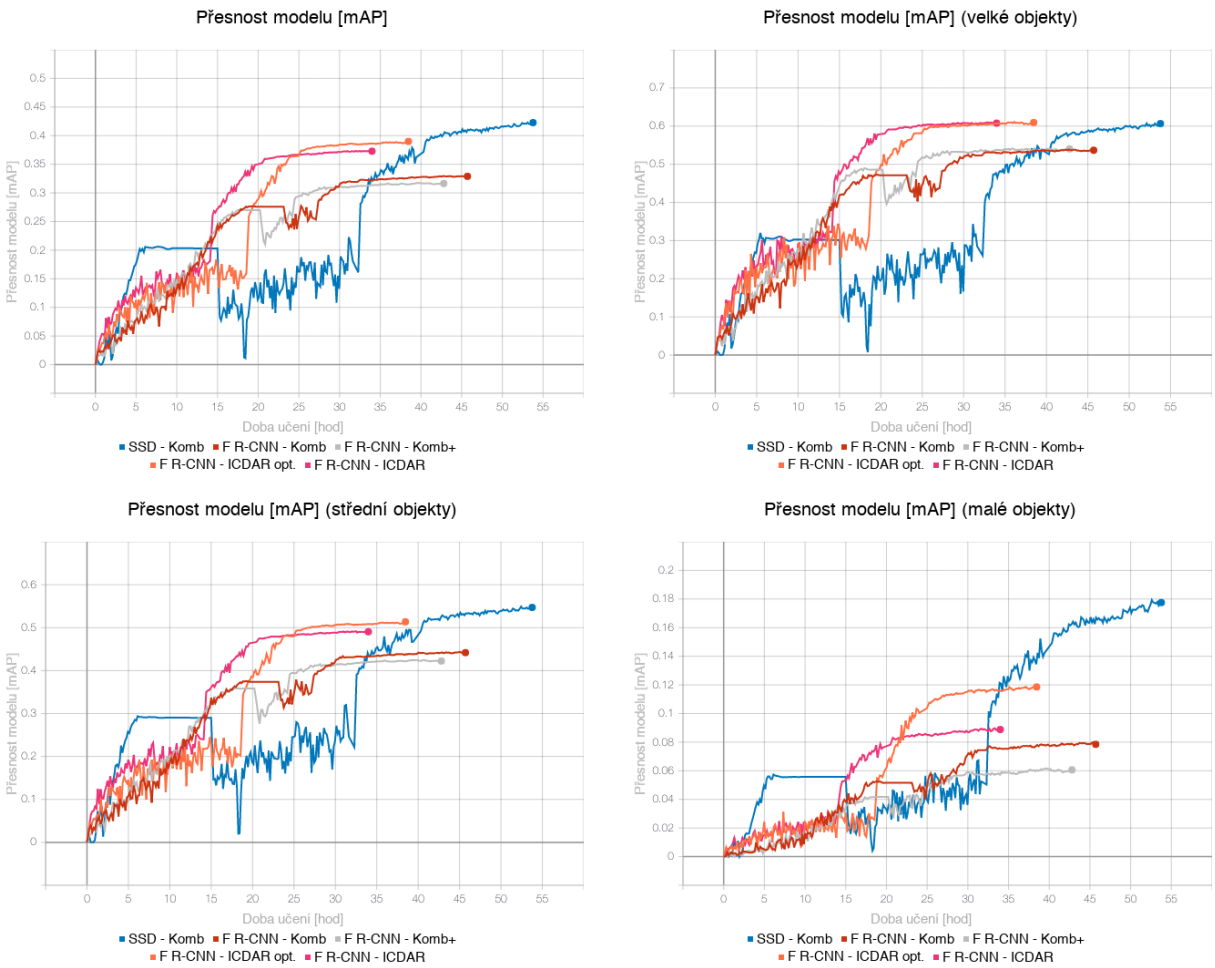
Tabulka 7: Vyhodnocení 5 nejlepších modelů pro všechny evaluační datasety. Poslední tabulky zobrazuje průměr modelů přes všechny evaluační sety. Kombinovaný+ znamená použití metod předzpracování pro umělé rozšíření kombinovaného datasetu. ICDAR opt. znamená dataset ICDAR s odstraněnými problémovými slovy (slova, která obsahují kromě žádaných jazyků take znaky latinky, či číslice) a vyváženými četnostmi každé třídy.

Evaluace na syntetickém datasetu		
Model	Trénovací set	mAP
SSD	Kombinovaný	89,96 %
F R-CNN	Kombinovaný	86,65 %
F R-CNN	Kombinovaný+	85,67 %
SSD	Syntetický	83,92 %
F R-CNN	Syntetický	78,78 %

Evaluace na kombinovaném datasetu		
Model	Trénovací set	mAP
SSD	Kombinovaný	67,51 %
F R-CNN	Kombinovaný	62,78 %
F R-CNN	Kombinovaný+	61,90 %
SSD	Syntetický	42,80 %
F R-CNN	Syntetický	41,11 %

Evaluace na reálném datasetu		
Model	Trénovací set	mAP
SSD	Kombinovaný	42,27 %
F R-CNN	ICDAR opt.	38,99 %
F R-CNN	ICDAR	37,28 %
F R-CNN	Kombinovaný	32,89 %
F R-CNN	Kombinovaný+	31,64 %

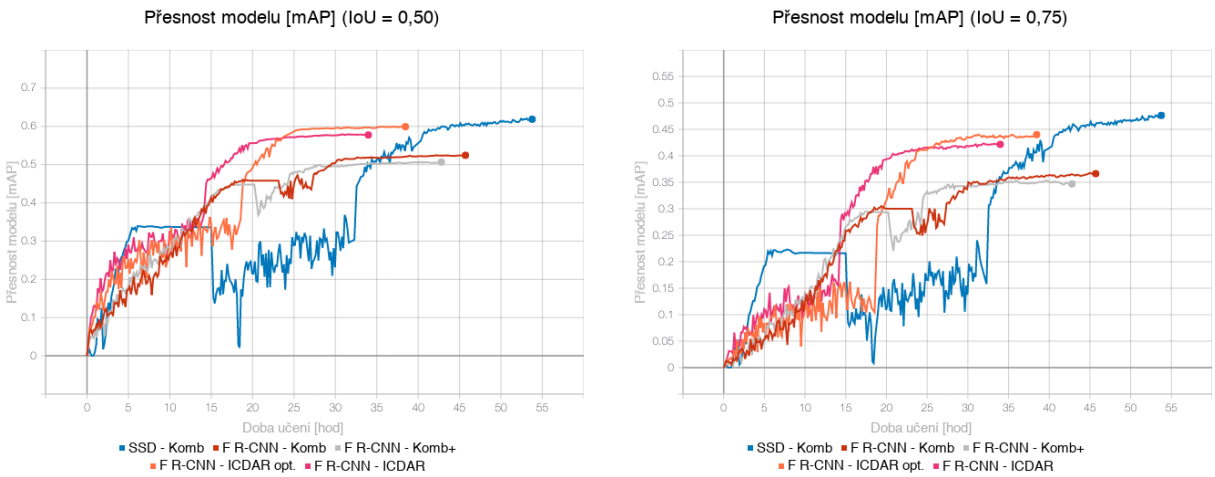
Průměr		
Model	Trénovací set	mAP
SSD	Kombinovaný	66,58 %
F R-CNN	Kombinovaný	60,77 %
F R-CNN	Kombinovaný+	59,74 %
SSD	Syntetický	42,57 %
F R-CNN	Syntetický	40,30 %



Obrázek 14: Vývoj mAP pěti nejlepších modelů dle rozměrů objektů. V legendě je vždy uveden typ modelu a dataset, na kterém byl učen. Rozdělení objektů na třetiny podle jejich velikosti umožňuje detailnější pohled na fungování modelu. Evaluační skóre se zde počítá vždy pro jednu skupinu objektů, přičemž objekty jiných velikostí se ignorují.

Obrázek (14) znázorňuje průběh učení modelů, jenž dosáhly nejvyšší přesnosti na reálném datasetu. Z tohoto grafu je zřetelně vidět neefektivní práce s učícím koeficientem. Na první pohled je zřejmé, že model od páté do patnácté hodiny učení stagnoval, a poté se jeho přesnost propadla téměř na nulu. V důsledku této chyby se teprve kolem třicáté hodiny učení vrátil na hodnotu přesnosti, které dosáhl už po pěti hodinách. Nebýt tohoto 25 hodinového zdržení, zdá se, že by tento model mohl konvergovat výrazně rychleji, než varianty Faster R-CNN.

Velmi zajímavé je chování modelů na malých objektech. Ty představují problém pro detektory objektů obecně. Pomocí kombinovaného datasetu se mi však podařilo natrénovat model, který zjevně tuto disciplínu zvládá o mnoho lépe. Z grafu přesnosti na malých modelech lze vyčíst, že druhou nejlepší variantu překonává o padesát procent, což je znatelný rozdíl, přestože se jedná o relativně nízké hodnoty. Kategorie velkých objektů je zde naprosto vyrovnaná. Na



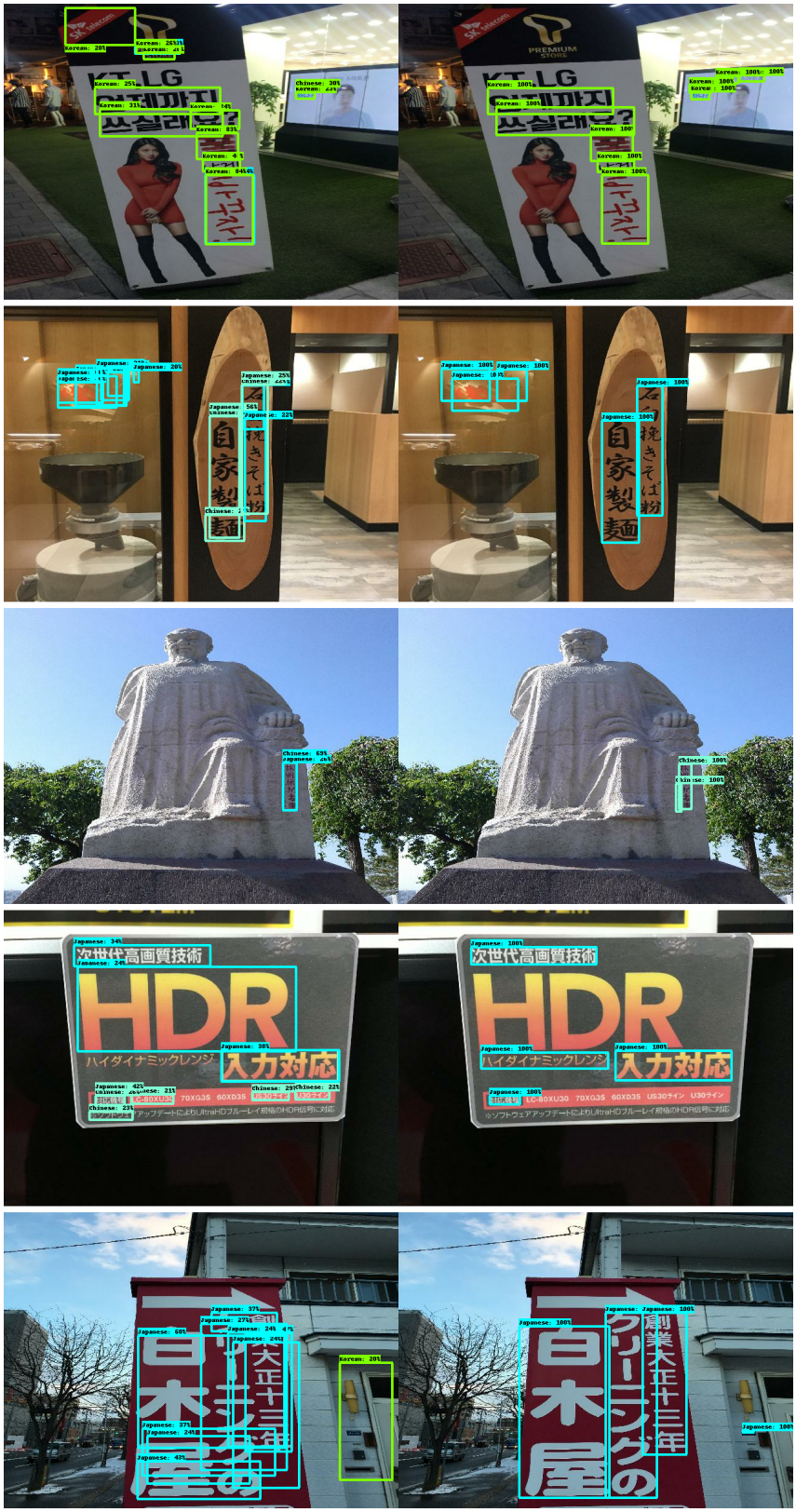
Obrázek 15: Vývoj mAP pěti nejlepších modelů dle IoU. Hodnota IoU 0,50 znamená, že detekce je označena jako správná pouze za předpokladu, že se její box překrývá s daným ground-truth boxem minimálně z padesáti procent.

objektech středních rozměrů rozdíl ve prospěch SSD činí zhruba 5 procentních bodů mAP. Zdá se tedy, že především na malých instancích získal daný model náskok, který mu zajistil prvenství i v celkové přesnosti.

Rovněž obrázek (15) nasvědčuje tvrzení, že vítězný model získává náskok především na komplikovanějších detekcích. Zatímco při hodnotě IoU nastavené na 0,5 (detekce je označena jako správná pouze za předpokladu, že se její box překrývá s daným ground-truth boxem minimálně z padesáti procent) je rozdíl mezi modely zanedbatelný, při stížení podmínek, tedy zvýšení hodnoty IoU na 0,75 se však náskok opět prohlubuje.

Výstupní data modelu při evaluaci reálného datasetu jsou vizualizovány na obrázcích (16) a (17). Jak lze na těchto obrázcích vidět, generuje model stále poměrně hodně falešných detekcí. V případě, že správně lokalizuje text, se už v klasifikaci zmýlí jen vyjímečně. Zajimává je schopnost modelu rozpoznávat instance velmi malých rozměrů, s takovými objekty mívají obvykle single-stage modely problémy.

Obrázek 16: Ukázka detekce modelu SSD. Obrázky vlevo jsou výstupem z modelu. Obrázky vpravo jsou ground-truth data evaluačního datasetu. Zelená barva značí korejštinu, azurová japonštinu a akvamarínová čínštinu.



Obrázek 17: Ukázka detekce modelu SSD. Obrázky vlevo jsou výstupem z modelu. Obrázky vpravo jsou ground-truth data evaluačního datasetu. Zelená barva značí korejštinu, azurová japonštinu a akvamarínová čínštinu.

6 Závěr

Podařilo se mi vytvořit detektor objektů, přizpůsobený pro detekci a identifikaci japonského, korejského, či čínského textu v obrázcích z reálného prostředí. Detektor je založen na modelu SSD[6], což je state-of-the-art model využívající single-stage architekturu. Vytvořil jsem dataset pro evaluaci extrahováním relevantních dat z obecného datasetu pro detekci textu ICDAR 2017 a 2019[30]. Na tomto datasetu o velikosti tisíc instancí na každou třídu jsem dosáhl výsledku 42,27 % mAP. V porovnání s obecnými datasety, kde zpravidla pouze pár vítězných metod překročí hranici 50 % [39][40][41] to považuji za uspokojivý výsledek.

Celkem jsem v rámci této práce učil 10 modelů, na kterých jsem testoval vliv použití různých konfigurací, architektur modelu, či datasetů. Celková doba učení byla 560 hodin. Pro rozšíření datasetu jsem vytvořil generátor syntetických dat, jehož výstupem jsou obrázky z reálného prostředí s vykresleným textem pro detekci a soubory obsahující metadata popisující objekty zájmu. Nejdříve jsem experimentoval s učením modelu pouze na takto vygenerovaných datech. Tyto pokusy mi potvrdily, že obecné modely pro detekci objektů založené na konvolučních neuronových sítích jsou zcela určitě schopny naučit se rozpoznávat i tak různorodé třídy jako jsou asijské jazyky obsahující desetitisíce znaků. Velmi rychle se mi podařilo naučit modely na přesnost přes 80 % při evaluaci proti umělému datasetu.

Poté jsem se zaměřil na zvyšování přesnosti při evaluaci proti reálným datům. Následnými experimenty jsem došel k závěru, že syntetická data nedokáží pro učení zcela nahradit data reálná. Model učený pouze umělými ani nezačal konvergovat. Na druhou stranu jsem prokázal, že spojením syntetických a reálných dat je možné výrazně zvýšit efektivitu učení a celkovou přesnost výsledného modelu. Obzvláště na složitějších instancích model učený na kombinovaném datasetu výrazně překonával modely naučené pouze na reálná data. Konkrétně až o padesát procent v případě detekce textů malých rozměrů.

Kromě generování vlastních dat jsem experimentoval také s rozšířením reálného datasetu pomocí různých metod předzpracování. Tento postup však nepřinesl žádné zlepšení modelu. Také jsem otestoval vliv kvality datasetu proti jeho velikosti. Odstraněním nekvalitních dat (textů, které obsahovaly znaky latinky, nebo číslice) a vyvážením četnosti zastoupení jednotlivých tříd v datasetu se mi podařilo mírně zvýšit přesnost výsledného modelu, řádově asi o pět procent.

Literatura

[1] R. Smith. An overview of the tesseract ocr engine. https://ieeexplore.ieee.org/abstract/document/4376991, Sep 2007. [Online; navštíveno 5.4.2018].

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, 2012. [Online; navštíveno 5.4.2018].

[3] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. https://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.html, June 2014. [Online; navštíveno 5.4.2018].

[4] Luca Maria Gambardella Jurgen Schmidhuber Dan Claudiu Ciresan, Ueli Meier. Convolutional neural network committees for handwritten character classification. https://ieeexplore.ieee.org/abstract/document/6065487, Nov 2011. [Online; navštíveno 5.4.2018].

[5] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster r-cnn: Towards real-time object detection with region proposal networks. https://arxiv.org/abs/1506.01497, 2015. [Online; navštíveno 5.4.2018].

[6] Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg. Wei Liu, Dragomir Anguelov. Ssd: Single shot multibox detector. https://arxiv.org/abs/1512.02325, Dec 2015. [Online; navštíveno 5.4.2018].

[7] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. https://arxiv.org/abs/1506.02640, 2015. [Online; navštíveno 5.4.2018].

[8] Anelia Angelova Joseph Redmon. Real-time grasp detection using convolutional neural networks. https://www.semanticscholar.org/paper/Real-time-grasp-detection-using-convolutional-Redmon-Angelova/07edaa3b52141dcd376a54490df3a91529f1bde3, 2015. [Online; navštíveno 5.4.2018].

[9] Schmidhuber Jürgen Ciresan Dan, Meier Ueli. Multi-column deep neural networks for image classification. https://arxiv.org/abs/1202.2745, June 2012. [Online; navštíveno 5.4.2018].

[10] Trevor Darrell Jitendra Malik Ross Girshick, Jeff Donahue. Rich feature hierarchies for accurate object detection and semantic segmentation. https://arxiv.org/abs/1311.2524, 2013. [Online; navštíveno 5.4.2018].

[11] Ross Girshick. Fast r-cnn. https://arxiv.org/abs/1504.08083, 2015. [Online; navštíveno 5.4.2018].

[12] T. Gevers A.W.M. Smeulders J.R.R. Uijlings, K.E.A. van de Sande. Selective search for object recognition. http://www.huppelen.nl/publications/selectiveSearchDraft.pdf, 2012. [Online; navštíveno 5.4.2018].

[13] Gang Yu Xiangyu Zhang Yangdong Deng Jian Sun Zeming Li, Chao Peng. Light-head r-cnn: In defense of two-stage object detector. https://arxiv.org/abs/1711.07264, 2017. [Online; navštíveno 5.4.2018].

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf, 2010. [Online; navštíveno 5.4.2018].

[15] D. Huttenlocher P. Felzenszwalb. Efficient graph-based image segmentation. http://cs.brown.edu/people/pfelzens/segment/, 2004. [Online; navštíveno 5.4.2018].

[16] Vaibhaw Singh Chandel. Selective search for object detection (c++ / python). https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/, 2017. [Online; navštíveno 5.4.2018].

[17] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, 2012. [Online; navštíveno 5.4.2018].

[18] V. Mach Learn Cortes, C. Vapnik. Support-vector networks. https://doi.org/10.1007/BF00994018, 1995. [Online; navštíveno 5.4.2018].

[19] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. http://arxiv.org/abs/1505.06798, 2015. [Online; navštíveno 5.4.2018].

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. http://arxiv.org/abs/1406.4729, 2014. [Online; navštíveno 5.4.2018].

[21] Yangqing Jia Pierre Sermanet Scott Reed Dragomir Anguelov Dumitru Erhan Vincent Vanhoucke Andrew Rabinovich Christian Szegedy, Wei Liu. Going deeper with convolutions. https://arxiv.org/abs/1409.4842, 2014. [Online; navštíveno 5.4.2018].

[22] Alexander Toshev Dragomir Anguelov Dumitru Erhan, Christian Szegedy. Scalable object detection using deep neural networks. https://arxiv.org/abs/1312.2249, 2013. [Online; navštíveno 5.4.2018].

[23] Dumitru Erhan Dragomir Anguelov Sergey Ioffe Christian Szegedy, Scott Reed. Scalable, high-quality object detection. https://arxiv.org/abs/1412.1441, 2015. [Online; navštíveno 5.4.2018].

[24] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. https://arxiv.org/abs/1512.03385, 2015. [Online; navštíveno 5.4.2018].

[25] Chen Sun Menglong Zhu Anoop Korattikara Alireza Fathi Ian Fischer Zbigniew Wojna Yang Song Sergio Guadarrama Kevin Murphy Jonathan Huang, Vivek Rathod. Speed/accuracy trade-offs for modern convolutional object detectors. https://arxiv.org/abs/1611.10012, 2017. [Online; navštíveno 5.4.2018].

[26] Qiang Huo Zhuoyao Zhong, Lei Sun. An anchor-free region proposal network for faster r-cnn based text detection approaches. https://arxiv.org/abs/1804.09003, 2018. [Online; navštíveno 5.4.2018].

[27] Ross Girshick Kaiming He Bharath Hariharan Serge Belongie Tsung-Yi Lin, Piotr Dollár. Feature pyramid networks for object detection. https://arxiv.org/abs/1612.03144, 2017. [Online; navštíveno 5.4.2018].

[28] Hanqiang Cao Olivier Rukundo. Nearest neighbor value interpolation. https://arxiv.org/abs/1412.1441, 2012. [Online; navštíveno 5.4.2018].

[29] Xiang Bai Xinggang Wang Wenyu Liu Minghui Liao, Baoguang Shi. Textboxes: A fast text detector with a single deep neural network. https://arxiv.org/abs/1611.06779, 2016. [Online; navštíveno 5.4.2018].

[30] S. Uchida M. Iwamura L. G. i. Bigorda S. R. Mestre J. Mas D. F. Mota J. A. Almazàn L. P. de las Heras D. Karatzas, F. Shafait. Icdar 2013 robust reading competition. https://ieeexplore.ieee.org/document/6628859/, 2013. [Online; navštíveno 5.4.2018].

[31] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. http://tensorflow.org/, 2015. [Online; navštíveno 5.4.2018].

[32] Derek Chow Chen Sun Menglong Zhu Matthew Tang Anoop Korattikara Alireza Fathi Ian Fischer Zbigniew Wojna Yang Song Sergio Guadarrama Jasper Uijlings Viacheslav Kovalevskyi Kevin Murphy Jonathan Huang, Vivek Rathod. Tensorflow object detection api. https://github.com/tensorflow/models/tree/master/research/object_detection, 2017. [Online; navštíveno 5.4.2018].

[33] Fernando Pérez Brian Granger Matthias Bussonnier Jonathan Frederic Kyle Kelley Jessica Hamrick Jason Grout Sylvain Corlay Paul Ivanov Damián Avila Safia Abdalla Carol Willing Jupyter Development Team Thomas Kluyver, Benjamin Ragan-Kelley. Jupyter notebooks – a publishing format for reproducible computational workflows. http://ebooks.iospress.nl/publication/42900. [Online; navštíveno 5.4.2018].

[34] Google Inc. Protocol buffers. https://developers.google.com/protocol-buffers/. [Online; navštíveno 5.4.2018].

[35] Google Inc. Tensorboard: Visualizing learning. https://www.tensorflow.org/guide/summaries_and_tensorboard. [Online; navštíveno 5.4.2018].

[36] Google Inc. Google colaboratory. https://colab.research.google.com/. [Online; navštíveno 5.4.2018].

[37] Google Inc. Google cloud platform. https://cloud.google.com/docs/. [Online; navštíveno 5.4.2018].

[38] Google Inc. Overview of the open images challenge 2018. https://storage.googleapis.com/openimages/web/challenge.html. [Online; navštíveno 5.4.2018].

[39] Serge Belongie Lubomir Bourdev Ross Girshick James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollár Tsung-Yi Lin, Michael Maire. Detection leaderboard. http://cocodataset.org/#detection-leaderboard. [Online; navštíveno 5.4.2018].

[40] Hao Su Jonathan Krause Sanjeev Satheesh Sean Ma Zhiheng Huang Andrej Karpathy Aditya Khosla Michael Bernstein Alexander C. Berg Olga Russakovsky, Jia Deng and Li Fei-Fei. Large scale visual recognition challenge 2017. http://image-net.org/challenges/LSVRC/2017/index#cite. [Online; navštíveno 5.4.2018].

[41] Jordi Pont-Tuset Matteo Malloci Jasper Uijlings Jake Walker Rodrigo Benenson Vittorio Ferrari, Alina Kuznetsova. Google ai open images - object detection track. https://www.kaggle.com/c/google-ai-open-images-object-detection-track/leaderboard. [Online; navštíveno 5.4.2018].